ZX Spectrum

(TS 2068)

Introducción al *Procesamiento de textos*

Randle Hurley



ZX SPECTRUM (TS 2068)
INTRODUCCION AL PROCESAMIENTO DE TEXTOS

CONSULTORES EDITORIALES
AREA DE INFORMATICA Y COMPUTACION

Antonio Vaquero Sánchez

Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid
ESPAÑA

Isaac Schnadower

Departamento de Electrónica
Universidad Autónoma Metropolitana
Gerente General de Servicios
Educativos Computacionales
MEXICO

Alfonso Pérez Gama

Ingeniero Electrónico
Universidad Nacional de Colombia
COLOMBIA

José Portillo

Universidad de Lima
PERU

ZX Spectrum (TS 2068)
INTRODUCCION AL PROCESAMIENTO DE TEXTOS

Randle Hurley

Traducción:
Alfonso Camaño Liceras
Licenciado en Ciencias Exactas

Revisión técnica:
Antonio Vaquero Sánchez
Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid

McGraw-Hill

MADRID • BOGOTÁ • BUENOS AIRES • GUATEMALA • LISBOA • MÉXICO
NUEVA YORK • PANAMÁ • SAN JUAN • SANTIAGO • SAO PAULO
AUCKLAND • HAMBURGO • JOHANNESBURGO • LONDRES • MONTREAL
NUEVA DELHI • PARÍS • SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY
TOKIO • TORONTO

ZX SPECTRUM (TS 2068). Introducción al Procesamiento de Textos

Atlacomulco, 499-501, Naucalpan de Juárez, Edo. de México
Miembro de la Cámara Nacional de la Industria Editorial, Reg. Núm. 465

ISBN: 968-451-731-9

Traducido de la primera edición en inglés de
The Spectrum Workshop-Word Processing and Beyond

ISBN: 0-07-084-704-5

Edición exclusiva para Ediciones La Colina, S. A. (España)

ISBN: 84-7615-011-3
Depósito legal: M. 36.401-1984

Compuesto en Fernández Ciudad, S. L. Catalina Suárez, 19. 28007 Madrid
Impreso en Gráficas EMA. Miguel Yuste, 27. 28037 Madrid

PRINTED IN SPAIN - IMPRESO EN ESPAÑA

A Kitty.

CONTENIDO

PROLOGO

El uso de programas sofisticados resulta divertido, tanto si se trata de programas de utilidades como si han sido escritos tan solo para entretenernos. Contemplar un procesador de textos en manos de un operador experto es casi como observar a un prestidigitador. Las cosas desaparecen para volver a aparecer más tarde en otro lugar. Pueden localizarse palabras particulares entre los miles de caracteres de un texto en tan solo unas décimas de segundo. Tan solo por diversión, podemos cambiar todas las 'f' del texto por 'ph'. No se habrá producido ningún daño permanente puesto que pueden volver a sustituirse en cuestión de pocos segundos. Del mismo modo es fascinante ver trabajar a un programa gestor de base de datos. Complicados conjuntos de datos confluyen en agrupaciones de acuerdo con la tarea que se tenga entre manos. Se necesita tiempo para asimilar el hecho de que la máquina reordena una colección de registros antes de que se haya podido levantar el dedo de la tecla. Es reconfortante utilizar este tipo de programa, sobre todo si el autor es uno mismo.

Usted puede crear estos programas por sí mismo y no tan solo burdas imitaciones de los programas comerciales. Los programas de este libro constituyen paquetes efectivos y potentes que pueden utilizarse para trabajar después de que la ilusión de la novedad se haya esfumado (esto es algo que no puede afirmarse de los programas de juegos). No es preciso ser un programador experto, el nivel de conocimientos adquiridos después de haberse leído el manual de la computadora será suficiente. Algunos de los capítulos de este manual son bastante complicados, pero no debe preocuparse si no los ha conseguido entender en su totalidad. Si hojea por un momento el libro se dará cuenta que no hay largos listados de programas. Los programas se han dividido en bloques manejables que son explicados uno por uno. Tampoco se precisan grandes porciones de memoria, por tanto si es necesario repetir alguna porción de programa no se origina un grave problema.

Este libro no es tan solo un conjunto de trozos a partir de los cuales pueden construirse los programas sin entender lo que se está haciendo. Lo verdaderamente divertido de trabajar en proyectos como éstos se alcanza cuando uno se ve a sí mismo analizando y comprendiendo cada trozo del programa y la forma en que se interrelacionan entre sí. Los programas han sido diseñados para ser introducidos incluso por los recién llegados al mundo de las computadoras y cada uno cuenta con un capítulo corto destinado a las instrucciones operativas y consejos. Esta información será de utilidad también para los lectores más avan-

zados. Se observará que hay suficiente información sobre los programas como para que puedan modificarse y adaptarse a necesidades especiales.

Siempre que es aconsejable se mezcla el código máquina con el BASIC de modo que en todo momento se utilice el mejor lenguaje. Los lectores que no han pasado del BASIC todavía se verán empujados a trabajar con un lenguaje más rápido y que se introduce con unas técnicas pedagógicas revolucionarias. Hay rutinas flexibles y potentes, cada una se acompaña con suficiente información como para que el lector la pueda incluir en alguno de sus programas. Esta misma idea persiste en el resto del libro, los programas están constituidos por bloques independientes que pueden utilizarse como componentes de otros proyectos.

Esta característica posibilita la construcción de programas más poderosos que sus propios componentes. Un ejemplo de esta técnica es el programa SPECHERGE. Este programa toma la salida de un procesador de textos y la ensambla con la lista de nombres y direcciones producida por otro programa para obtener un conjunto de cartas iguales pero destinadas a personas diferentes.

Sin embargo, aunque utilice los programas tal y como son (sin combinar componentes para formar programas más complicados) le resultarán entretenidos. La utilidad de los programas asegura su larga duración y la comprensión que puede alcanzar al trabajar en estos proyectos le habilitarán para producir sus propios programas de alta calidad.

RANDLE HURLEY

Nota del editor

Los listados de los programas de este libro se han impreso directamente desde el Spectrum, utilizando para ello una impresora Epson MX 100 III. Todos los listados están comprobados, pero el lector debe tener cuidado de no confundir la letra minúscula 'L' (`l`) con la cifra 'uno' (`1`), particularmente en los programas en lenguaje ensamblador, donde `ld a,l` se parece a `ld a,1` pero los resultados son absolutamente distintos.

Nota del traductor

La traducción del libro ha sido realizada en su mayor parte con la ayuda de un procesador de textos para el ZX Spectrum similar a los que se estudian en este libro. El ahorro de tiempo y la comodidad que ha significado este procedimiento son muy dignos de tenerse en cuenta.

1 DESCRIPCION DE LOS PROGRAMAS

La falta de imaginación de las computadoras impide que sufran equivocaciones como las personas al realizar tareas tediosas y repetitivas. Las computadoras no se aburren ni se cansan y como consecuencia pueden prestar toda su atención al mismo trabajo durante tanto tiempo como sea necesario. Además las computadoras son muy rápidas, las personas por el contrario trabajan a una velocidad muy modesta. A causa de su imaginación rápidamente se aburren con tareas repetitivas y su nivel de concentración decae en gran medida. En este libro se incluyen un conjunto de programas que utilizan de la mejor manera las capacidades de las personas y de las máquinas. La computadora puede trabajar sin pensar, solicitando nuevas instrucciones cuando sea necesario. Si tardamos en darle nuevas instrucciones la máquina esperará pacientemente como para permitirnos que nuestra imaginación pueda continuar funcionando. La relación hombre-computadora funciona porque los talentos de los dos son complementarios.

El procesador de textos es un ejemplo interesante de cómo conjuntar los talentos del hombre y de la máquina. La característica esencial de un procesador de textos es que se introduce un retraso entre la pulsación de las teclas y la impresión del texto en papel. El retraso puede ser tan largo como quiera el usuario y mientras tanto puede ser editado hasta que sea perfecto. Si después de la impresión resulta que la perfección era un puro espejismo, puede repetirse el proceso de edición para obtener una nueva versión. Un procesador de textos puede ser descrito como una máquina de escribir electrónica que permite correcciones, escribe sobre la pantalla y aguarda instrucciones antes de reproducir su contenido en papel.

Muchos procesadores de textos realizan mucho más que esta tarea básica. Hay dos programas de procesadores de textos en los capítulos que siguen. En el primero se identifican las tareas básicas y las técnicas necesarias para llevarlas a cabo. El programa está completamente escrito en BASIC y el resultado es un procesador de textos que cumple perfectamente las especificaciones dadas anteriormente.

Hay un segundo procesador de textos que va mas allá de los

requerimientos mínimos. Este programa está basado en los fundamentos desarrollados por el anterior. El programa escrito en BASIC tuvo que detenerse allí donde la velocidad debía ser superior a la proporcionada por el intérprete BASIC. El segundo programa utiliza la velocidad que proporciona el código máquina para salirse del marco de las especificaciones mínimas. El resultado es un paquete flexible y de alta velocidad que puede manipular tres veces mas texto que la versión anterior. El contenido de este libro ha sido escrito en el programa avanzado y la velocidad de producción ha sido superior a la de libros anteriores. No fue necesario escribir de nuevo páginas si alguna idea necesitaba modificarse. Los cambios se realizaban en la pantalla y la computadora se encargaba de recolocar y reimprimir el texto sobre papel. Esto no significa que la máquina de escribir haya perdido su utilidad, la computadora se conectó a la máquina y la escritura se realizó en ésta. Como consecuencia la copia quedó mucho más vistosa.

El lenguaje máquina es tan complicado como útil. Este es un área que todavía padece la limitación que supone que sólo los expertos puedan conocer sus secretos y traten de preservarlos. El arma utilizada para preservar estos secretos es la jerga que utilizan. Concentrando las ideas (instrucciones) en palabras de tres letras se consideran suficientemente protegidos excepto de los intrusos más obstinados. En el libro se utiliza una nueva técnica de aproximación al lenguaje máquina y los resultados son muy satisfactorios. El 'código máquina' se escribe con un número reducido de instrucciones BASIC. Solo se utilizan las instrucciones que tienen un equivalente en lenguaje máquina. Los programas pueden ser comprobados y corregidos en BASIC utilizando todas las facilidades de la computadora. Cuando se concluye la corrección los programas se traducen a código máquina con un esfuerzo mínimo. Todos los detalles sobre este nuevo procedimiento están en el Capítulo 6.

SPECFILE es más que un sencillo sistema de ficheros. Es un programa que le permitirá obtener más de su fichero de lo que pudo parecerle posible cuando almacenó la información. Existen otras ventajas derivadas del uso del programa que pueden no ser obvias al principio. Los ficheros que se producen ocupan poco espacio y se necesitan tan solo unos pocos minutos para producir una copia de seguridad. Cuando se introducen nuevos registros, no es necesario hacerlo en orden; la máquina puede ordenarlos en el tiempo que tardaríamos en pulsar unas pocas teclas.

En el corazón del programa está una rutina en lenguaje máquina que permite la clasificación. La clasificación es rápida, pero ésta no es la principal fuente de poder. Los registros de un fichero

pueden clasificarse sobre la base de 15 subregistros. Si se están llevando a cabo varias clasificaciones de diferentes subregistros, una detrás de otra, entonces pueden imprimirse conjuntos de registros que satisfagan interesantes combinaciones de características. Esto posibilita el acceso a cualquier parte de la información contenida en el fichero. El uso de esta capacidad es tan variado que se ha escrito un capítulo entero para sugerir algunos de ellos. SPECFILE no es sólo potente es también flexible. El programa se rediseñará a sí mismo para que su almacenamiento se acomode exactamente a las necesidades de los registros. La entrada de datos se realiza rellenando un 'formulario' sobre la pantalla.

Una aplicación de gran importancia para muchas personas aparece en el capítulo sobre SPECMERGE. Este programa permite concatenar la salida de un procesador de textos con una lista de correo producida por SPECFILE. Se utilizan marcadores especiales para intercalar texto de los registros entre el texto de la carta. El resultado es una serie de cartas cada una ligeramente diferente. En cada dirección se recibe por tanto una carta que ha sido especialmente preparada para él o ella de acuerdo con el contenido del fichero.

Las rutinas en lenguaje máquina están en dos bloques cada una de los cuales contiene tres de ellas. Una amplia documentación hace que las rutinas sean trasportables. Puesto que no se hacen llamadas a las rutinas de la memoria ROM, los programas pueden utilizarse en otras computadoras basadas en el microprocesador Z-80 con pocas o ninguna modificación. Todos los programas se presentan siempre con la misma idea. Siempre se dispone de suficiente información como para permitir que las ideas vertidas en los diferentes capítulos sean usadas lo más ampliamente posible.

Algunos de los capítulos se han incluido tan sólo como referencia. Para evitar que quienes únicamente buscan las instrucciones operativas tengan que leerse el resto del libro, la información de algunos capítulos es algo repetitiva. En un libro tan denso como éste, la repetición no es tan mala. Los programas se presentan en pequeños bloques de unas pocas líneas. Cada bloque cumple una función que se explica en el mismo párrafo. Si surge la misma función otra vez en el programa un poco más tarde, de nuevo se repite la explicación. Si las personas tuvieran la capacidad de memoria de las computadoras, estas repeticiones serían innecesarias, pero tampoco sería necesario el uso de estas máquinas.

Los desarrollos de programas nunca se acaban. Al igual que un cuadro, siempre pueden perfeccionarse, siempre caben mejoras en los programas y los que se incluyen en este libro no son una

excepción. Sería arrogante afirmar lo contrario. Mientras que el libro se ha editado han surgido nuevas ideas que mejoran los programas. No ha sido posible añadirlas en el momento de la escritura pero sería ilógico que no se pudiera disponer de ellas. La cinta cassette diseñada para acompañar este libro continuará siendo actualizada y desarrollada de manera que en todo momento contenga la versión más potente y actual.

2 SPECPRO. EL PROCESADOR DE TEXTOS EN BASIC

La computadora con la que estamos trabajando es una máquina muy flexible. Mientras que otras computadoras limitan el número máximo de caracteres de una cadena a unos pocos cientos, en ésta se pueden manipular cadenas de miles de caracteres. Estas cadenas enormes pueden dividirse utilizando el procedimiento poco ortodoxo del sistema para manipular texto; los cambios así producidos son tan rápidos como si su control se realizase en lenguaje máquina. La consecuencia es que el proceso de editar texto es mucho más simple de lo que podría esperarse. Las rutinas de edición ocupan tan solo 18 líneas del programa.

Un procesador de textos tiene que realizar un número mínimo de tareas fundamentales. Obviamente debe almacenar el texto, pero también debe permitir la corrección de errores, el borrado de parte del texto y la inserción de forma simple. Cuando se hayan realizado cambios no debe quedar rastro del texto anterior. Cuando el texto haya sido editado a satisfacción del escritor, la impresora debe ponerse en marcha con solo pulsar algunas teclas. El producto de la impresora debe estar 'justificado', es decir, cada línea debe empezar con una nueva palabra a excepción de los comienzos después de un punto y aparte. El escritor debe poder especificar cuando termina una línea o una página y antes de ordenar la impresión debe poder examinar el aspecto sobre el papel. Tiene que ser posible desconectar la impresora y examinar todo el texto sobre la pantalla. Como es posible que se vaya a reutilizar el texto en otra ocasión es necesario que se pueda grabar el contenido del procesador de textos en cinta y, por supuesto, que se pueda cargar de nuevo en la computadora.

SPECPRO, el procesador de textos en BASIC, realiza todas estas tareas de forma rápida y eficiente. Las técnicas utilizadas en este programa constituyen los fundamentos en los que se basan los procesadores más complicados escritos en lenguaje máquina. Seguir este primer programa facilita el camino hacia SPECTEXT que es un programa más avanzado. Puede resultar incluso divertido observar como se construye el programa y si se entiende, la comprensión de SPECTEXT será mucho más fácil. El estilo de presentación que se ha adaptado facilita la comprensión. Cada idea se presenta individualizada para que gradualmente se alcan-

ce una visión de conjunto ayudándose de una cuidadosa documentación.

Antes de comenzar a analizar las líneas del programa es conveniente dar una idea general de cómo es el programa. Los caracteres que se teclean aparecen en la pantalla antes de nada, pero también deben ser recordados por la computadora. La pantalla es tan sólo un almacenamiento temporal, el almacenamiento real se realiza en lo que se denomina 'buffer' del texto que es una cadena de caracteres que se dimensiona mediante DIM para 8400 caracteres. Deben utilizarse como máximo 8000 espacios para el almacenamiento de caracteres, esto significa que alrededor de 1500 palabras pueden ser manipuladas por la computadora. Los procesadores más avanzados pueden manipular cantidades muy superiores porque al estar escritos en lenguaje máquina no ocupan tanta memoria. Se pueden manipular documentos más largos si se graban por partes en cinta según van siendo escritos.

Este programa se ha escrito alrededor de una lista de opciones que se denomina 'menú'. Todas las rutinas retornan a dicho menú excepto la que permite salirse del programa. El programa comienza con una serie de subrutinas seguidas del menú. El menú se ha diseñado de tal forma que el resto de las partes del programa se agrupan en bloques separados cada uno por 1000 líneas. Esto facilita al usuario el añadir nuevas posibilidades al programa o el modificar las ya existentes adaptándolas a sus necesidades. La rutina más importante es la que controla la entrada de texto, su almacenamiento y su escritura en la pantalla. Esta rutina comienza en la línea 1000. La rutina que permite cargar el material almacenado para editarlo comienza en la línea 4000. El programa permite al usuario cambiarse fácilmente de entrada a edición sin que tenga que volver al menú.

Muchos procesadores de textos se controlan mediante teclas especiales que el usuario tiene que conocer antes de ser capaz de utilizar el programa eficazmente. SPECPRO dispone de un conjunto de comandos que permiten al usuario manejar el programa utilizando tan solo dos teclas, 'Z' y ENTER. Si se necesitasen otras teclas la computadora imprimirá mensajes recordando que tecla pulsar para obtener la acción que se desee. Un programa difícil de usar acabará por no utilizarse.

Las dos rutinas que manipulan la cinta se encuentran en las líneas 2000 y 5000. La información previamente grabada puede cargarse pulsando la opción '2' del menú mientras que la opción '5' permite grabar todo el contenido de la computadora para su uso futuro. La rutina de formato de la línea 3000 permite la presentación del texto en la pantalla del mismo modo en que aparecerá en papel. Para aprovechar de la mejor manera el espacio

disponible los finales de párrafo, línea y página se indican en el texto mediante marcadores especiales. La rutina de formateo interpreta correctamente dichos marcadores imprimiendo los espacios, líneas en blanco y números de página. Es esta rutina la que se asegura de que cada línea comienza con una nueva palabra y la que espacia las líneas ajustándolas adecuadamente. La decisión de enviar o no enviar el texto a la impresora depende de las respuestas que se den a las preguntas que formula la rutina de mantenimiento.

La rutina de mantenimiento permite conservar el orden en la computadora y conectar o desconectar la impresora. Si la computadora está conectada a la impresora, necesitará conocer el número de líneas por página que se quiere imprimir, así como el número de líneas en blanco antes de una nueva cabecera de página.

Cuando el programa se carga por primera vez en la computadora es preciso llamar en primer lugar a la rutina de mantenimiento en donde se inicializa toda la información que necesita la máquina para su trabajo. También es necesario redefinir uno de los caracteres gráficos de tal forma que puedan utilizarse las comillas sin interferir el manejo que hace la computadora de las cadenas. Todo esto se lleva a cabo grabando el programa de manera que se autoejecute en la línea 61ØØ y no en el menú. Tan pronto como la rutina de mantenimiento cumple con su cometido, aparece el menú para que el usuario empiece a trabajar. Será necesario desconectar la impresora para comenzar un nuevo texto o para cambiar la longitud de la página, en ese momento una nueva llamada a la rutina de mantenimiento permitirá llevar a cabo cualquiera de estos objetivos.

EL LISTADO DEL PROGRAMA

Desde casi el principio del programa comienzan las subrutinas. Para evitar que el usuario caiga en ellas por accidente, la línea 1 las salta enviando el control a la línea 9ØØ que es donde se imprime el menú. La primera de las subrutinas lee el teclado y trasfiere el carácter que se esté tecleando a la variable b$. La línea 1Ø se asegura de que se ha dejado de pulsar la última tecla que se hubiera presionado. La línea 11 espera otra vez que se pulse una tecla. Una vez que se ha almacenado un carácter en la variable b$, la línea 12 produce un sonido que indica al usuario que la tecla pulsada ha sido interpretada. Después de este sonido, la máquina vuelve al programa principal.

```
  1 GO TO 900
  10 IF INKEY$ <>"" THEN GO TO 1
0
  11 LET b$=INKEY$: IF b$="" THE
N  GO TO 11
  12 BEEP .003,0: RETURN
```

Hay dos subrutinas más que se verán más adelante. La siguiente sección que se va a describir es el menú de opciones. Hay una instrucción de limpieza de pantalla CL5 al principio y al final de esta rutina. La línea 900 imprime la mayor parte del texto. Tres espacios separan cada número de la descripción de la opción que representa para mejorar la calidad de la imagen. La línea 910 lee el teclado y comprueba el carácter que se haya pulsado. Sólo se aceptan los números comprendidos entre 1 y 7, cualquier otro carácter obliga al usuario a pulsar de nuevo. La línea 920 prepara la pantalla para la escritura si se ha elegido la opción 1. El cursor que se imprime en la línea 21, columna 0, indica dónde aparecerá el primer carácter que se pulse. El segundo 'AT 21,0;' obliga a la computadora a imprimir sobre esa línea. La última línea de esta rutina envía el control a un número de línea de acuerdo con la opción del menú que se haya seleccionado. El efecto de esta línea 930 es que obliga a agrupar las diferentes rutinas en bloques de 1000 líneas.

```
 900 CLS : PRINT "    SPECTRUM W
ORD PROCESSOR "''''"1   Enter tex
t"''"2   Load text from tape"''"3
  Print text"''"4   Read through/
edit text"''"5   Save text on tap
e"''"6   Organise system for work
"''"7   Stop work"
 910 GO SUB 10: IF b$ <"1" OR b$ >
"7" THEN GO TO 910
 920 CLS : IF b$="1" THEN  PRINT
 AT 21,0;"_";AT 21,0;
 930 GO TO VAL b$*1000
```

Lo primero que tiene que hacer la computadora cuando se está introduciendo texto es leer el teclado. La línea 1000 cumple esta función y comprueba después que 'c' (la variable que apunta hacia la siguiente posición en el texto) no ha llegado al final del almacenamiento. La línea 1500 imprime el mensaje 'BUFFER FULL' que significa que se ha agotado la capacidad de almacenamiento. Los caracteres cuyos códigos son inferiores a 32 o supe-

riores a 127 son caracteres de control y no pueden ser impresos. De estos caracteres nos ocuparemos más tarde, por el momento la línea 1010 los filtra. La línea 1020 coloca el carácter en el lugar adecuado del almacenamiento del texto, a$, le imprime en la pantalla y actualiza después todos los contadores que indican a la máquina dónde está en todo momento. La variable 'c' apunta hacia el siguiente lugar del almacenamiento del texto y 'b' apunta hacia la siguiente columna en la pantalla. Cuando 'b' alcanza el valor 33 pasa a valer de nuevo 1. La sentencia POKE de la línea hace que el texto se traslade hacia arriba automáticamente sin necesidad de preguntar. De esta manera las operaciones del programa se hacen más cómodas. La línea 1030 cierra el bucle.

```
1ØØØ GO SUB 1Ø: IF c< 8ØØØ THEN
GO TO 15ØØ
1Ø1Ø IF CODE b$<32 OR CODE b$>12
7 THEN  GO TO 1Ø4Ø
1Ø2Ø LET a$(c)=b$: PRINT b$;: LE
T b=b+1: LET c=c+1: IF b=33 THEN
  LET b=1: POKE 23692,9
1Ø3Ø GO TO 1ØØØ
```

El control del programa se traslada a la siguiente sección sólo si el código del carácter que se acaba de pulsar está fuera del rango normal. Dichas pulsaciones pueden tener su origen en errores o por el contrario pueden tener significados especiales. La línea 1040 da a la variable 'd' el valor de un número de línea adecuado. La línea 1050 filtra los caracteres a excepción de DELETE cuyo código es 12. La línea 1060 borra el último carácter de la pantalla y resta uno de los contadores. La siguiente línea imprime un cursor, lee un nuevo carácter y vuelve al bucle principal. La línea 1080 rechaza todos los caracteres que pudieran causar permitiendo solo ENTER para pasar a la siguiente sección.

```
1Ø4Ø LET d=21
1Ø5Ø IF CODE b$<>12 THEN  GO TO
1Ø8Ø
1Ø6Ø LET b=b-1: LET a$(c)=" ": L
ET c=c-1: IF b=Ø THEN  LET b=32:
 LET d=d-1: PRINT " ";AT 19,Ø;
1Ø7Ø PRINT AT d,b-1;"_ ";AT d,b-
1;: GO SUB 1Ø: GO TO 1Ø5Ø
1Ø8Ø IF CODE b$<128 AND CODE b$>
31 THEN  GO TO 1Ø1Ø
```

```
1Ø9Ø IF CODE b$<>13 THEN GO TO
1ØØØ
```

La línea 11ØØ imprime un pequeño menú en la parte inferior de la pantalla que sirve como recordatorio de las teclas que deben pulsarse para indicar a la computadora dónde comenzar nuevas líneas, párrafos y páginas. También se incluye en esta línea un filtro que envía el control a la rutina de edición de la línea 4ØØØ. Las siguientes líneas interpretan las instrucciones dadas por el usuario tomando las medidas apropiadas. Se preparan los punteros antes de retornar al bucle principal en la línea 1ØØØ.

```
11ØØ INPUT "1line 2para 3page (Z
/Enter)";b$: IF LEN b$<>1 OR(b$
<>"z" AND (b$<"1" OR b$>"3"))TH
EN  GO TO 4ØØØ
111Ø IF b$="z" THEN  GO TO 9ØØ
112Ø LET c$=CHR$ 131: IF b$="2"
THEN  LET c$=CHR$ 139
113Ø IF b$="3" THEN  LET c$=CHR$
 134
114Ø LET a$(c)=c$: LET c=c+1: LE
T b=b+1: PRINT c$;: IF b>31 THEN
  LET b=b-31
115Ø GO TO 1ØØØ
```

Cuando se satura el almacenamiento se ejecuta la siguiente sección.

```
15ØØ BEEP .1,2Ø: PRINT AT 9,9;"
BUFFER FULL  ";TAB 9;" FILE ON T
APE ": IF INKEY$<>"z" THEN GO T
O 15ØØ
151Ø GO TO 9ØØ
```

Hemos concluido una de las secciones más largas de este programa. Más tarde se añadirán un par de segmentos. El bloque que sigue es corto y directo. Posibilita cargar textos pregrabados en la computadora. Durante la ejecución se redimensiona la cadena, de este modo se limpia totalmente. Las líneas 2ØØØ y 2Ø1Ø leen el nombre del texto que se pretende cargar y ofrecen una oportunidad de volver al menú. Esto es importante, puesto que las opciones del menú no responden siempre a lo que el usuario pretende. La línea 2Ø2Ø inicializa todos los punteros al principio del texto. El texto comienza en el carácter número 321 de la cadena, con

objeto de suavizar las operaciones del procesador de textos. El uso de la primera parte de la cadena haría difícil la edición de la primera media página. No se preocupe en detalle de la instrucción de carga que figura en la línea 2030 puesto que está de acuerdo con la opción 5 del menú que se refiere a la grabación de texto en cinta. No es necesario crear etiquetas, puesto que la computadora las genera automáticamente. Todas las instrucciones se imprimen en la pantalla cuando se ejecuta el programa.

```
2000 PRINT "Key the name of the
section of text on the tape ":
INPUT b$: PRINT 'b$
2010 PRINT '"key C to confirm":
INPUT c$: IF c$ <> "c" THEN GO TO
 900
2020 DIM a$(8400): LET p=0: LET
b=1: LET c=321: POKE 23692,9:
2030 PRINT '"find correct sectio
n of tape": INPUT "key ENTER ";c
$: LOAD b$ DATA a$()
2040 GO TO 900
```

La siguiente opción es la más compleja de todo el programa. Una vez finalizada esta parte, el resto del programa es muy simple. En esta sección se organiza el texto para ser impreso. El aspecto ofrecido por el texto sobre el papel es de suma importancia, por este motivo cada línea empieza con nueva palabra y éstas no se parten entre dos líneas. Se interpretan todos los marcadores y se añaden los espacios necesarios para mejorar el aspecto de la página. En esta sección se utilizan gran cantidad de nuevas variables. Al final del capítulo se ofrece una lista de todas las variables y sus usos. Por el momento, todo lo que necesita recordar es que 'x' apunta al siguiente carácter que va a ser impreso e 'y' contiene el número de caracteres por línea más uno. En la línea 3030 se comprueba el código de todos los caracteres para ver si se trata de marcadores. Todos los marcadores tiene códigos por encima de 127. De ellos nos ocuparemos más adelante. La línea 3050 devuelve el control al menú si se ha concluido la impresión. Las dos líneas siguientes cargan en b$ una línea completa de caracteres para localizar entre ellos marcadores. Todos ellos marcan el final de una línea, por ello cuando se encuentra alguno, la variable 'y' toma el valor de la posición del marcador; 'y' es el último carácter que se imprimirá en la línea. Las líneas 3080 y 3090 vuelven a comprobar los espacios de la línea. Si no hubiera ninguno no es preciso realizar ninguna acción, por lo que pueden saltarse el res-

to de las comprobaciones para incrementar la velocidad del proceso de formateo.

```
3ØØØ LET x=321
3Ø1Ø LET x1=1: LET x2=x1: LET x4
=x1
3Ø2Ø LET y=33
3Ø3Ø LET f=CODE a$(x): IF f>127
THEN  LET x=x+1: IF f<>131 THEN
 GO TO 32ØØ
3Ø5Ø IF x+y>LEN a$ THEN  GO TO 9
ØØ
3Ø6Ø LET b$=a$(x TO x+y): LET f=
Ø: FOR j=1 TO y+1: IF CODE b$(j)
>127 AND f=Ø THEN  LET f=j
3Ø7Ø NEXT j: IF f>Ø THEN LET y=
f
3Ø8Ø LET f=Ø: FOR j=1 TO y: IF b
$(j)=" " THEN  LET f=1: LET j=y
3Ø9Ø NEXT j: IF f=Ø THEN  GO TO
311Ø
```

El siguiente trabajo que se necesita realizar es ajustar la línea a su última palabra completa. La línea 31ØØ mueve 'y' un espacio hacia atrás si el carácter no es un espacio o un marcador. La subrutina de la línea 2Ø realiza algunas cosas más que la impresión. La línea 312Ø escudriña el teclado buscando una 'Z' que es la señal de escape y devuelve el control al menú cuando la encuentra. La subrutina de la línea 34ØØ comprueba si se ha completado una página en cuyo caso se imprime un número de línea y algunas líneas en blanco como cabecera de la página siguiente. Esta rutina retorna a la línea 3Ø2Ø donde se inicializa la longitud de una línea a 33 caracteres.

```
31ØØ IF b$(y)<>" " AND CODE b$(y
)<128 THEN  LET y=y-1: GO TO 31Ø
Ø
311Ø GO SUB 2Ø: IF CODE b$(y)>12
7 THEN LET x=x-1
312Ø IF INKEY$="z" THEN  GO TO 9
ØØ
313Ø GO SUB 34ØØ
314Ø GO TO 3Ø2Ø
```

Ahora conviene comentar la subrutina de la línea 2Ø. La primera línea imprime el texto ya preparado y mueve después 'x' al

siguiente carácter por imprimir. Si la computadora sabe que la impresora está conectada entonces 'p' toma el valor 1 en lugar de su otro posible que es Ø. Si se han impreso 22 líneas y 'p' tiene el valor 1 se habrá reproducido la pantalla. De esta forma la computadora sabe que puede limpiar la pantalla para que sea ocupada por el siguiente bloque del texto. La siguiente línea suma uno al número de líneas impresas en la pantalla desde que se limpió y uno al número de líneas impresas en papel desde que se completó la última página de texto. Cuando se han arreglado todos los contadores el control vuelve al programa principal.

```
  2Ø PRINT b$( TO y-1): LET x=x+
y
  21 IF x1=22 AND p=1 THEN  COPY
 : CLS : LET x1=Ø
  22 IF x1=22 AND p=Ø THEN  INPU
T c$: CLS : LET x1=Ø
  23 LET x1=x1+1: LET x4=x4+1: R
ETURN
```

La variable 'f', como se recordará, contiene el valor del código del marcador que debe ser interpretado por la computadora. Los dos marcadores que quedan por examinarse, nuevo párrafo y nueva página, implican ambos la impresión de líneas en blanco. Cada vez que se imprime una nueva línea, se comprueban los contadores para ver si se ha acabado la pantalla o la página. La subrutina de la línea 2Ø hace todas las comprobaciones necesarias, pero como no se necesita imprimir nada, se salta la primera línea. Este es el motivo para las diversas llamadas a subrutina línea 21, que se hacen mediante GO SUB en los dos bloques del programa que siguen. Hay otra subrutina que examina la impresión y que está en la línea 34ØØ. Esta rutina prepara la impresión del número de página en la parte inferior de cada una cuando están completas. Las líneas 32ØØ y 323Ø imprimen los espacios necesarios para indicar el principio de un párrafo y hacen todas las comprobaciones necesarias.

```
32ØØ IF f<>139 THEN  GO TO 33ØØ
321Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: LET a=x3-x4: IF a>3 THEN F
OR j=1 TO a: PRINT : GO SUB 21:
GO SUB 34ØØ: NEXT j: GO TO 323Ø
322Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: PRINT : GO SUB 21: GO SUB 3
4ØØ
```

```
3230 PRINT TAB 5;: LET y=28: GO
TO 3030
```

Si el marcador no hubiera sido de fin de párrafo entonces la línea **3200** del último bloque del programa habría mandado el control a la sección que ahora vamos a analizar. La línea **3300** obliga a la computadora a detenerse si se detecta algún código inválido. Aunque no debería haber ningún marcador esta línea los tiene en cuenta por si se deslizasen. Las líneas comprendidas entre la **3300** y la **3320** imprimen suficientes espacios como para completar la página. Cuando la página está completa, se añade el número de página y se comienza una nueva.

```
3300 IF f<>134 THEN PRINT AT 10
,6;" INVALID CHARACTER ": INPUT
b$: GO TO 900
3310 LET a=x3-x4: PRINT : GO SUB
 21: FOR j=1 TO a: PRINT : GO SU
B 21: NEXT j: GO SUB 3410
3320 GO TO 3020
3400 IF x4<x3 OR p=0 THEN RETUR
N
3410 FOR j=1 TO 5: PRINT : GO SU
B 21: IF j=3 THEN  PRINT TAB 15;
x2: LET x2=x2+1: GO SUB 21
3420 NEXT j: LET x4=1: RETURN
```

Con estos cortos segmentos se concluyen las rutinas de formateado e impresión. La siguiente parte del programa trata de la edición del texto. La variable ‘c’ se conoce como el puntero del texto, puesto que apunta hacia la posición del carácter en proceso. La primera tarea consiste en comprobar que el puntero del texto señala una de las partes sensibles de buffer. La línea **4020** imprime la parte correcta del texto en la pantalla, de tal forma que el carácter al que apunta ‘c’ esté al principio de la línea **10** de la pantalla. Las variables ‘m’ y ‘n’ son las coordenadas en la pantalla de esta posición y se utilizarán para imprimir un cursor. Al concluir se envía a la computadora a la subrutina que permite al usuario mover el cursor.

```
4000 IF c<321 THEN  LET c=321
4010 IF c>LEN a$-320 THEN LET c
=LEN a$-319
4020 CLS : PRINT a$(c-320 TO c+3
19): LET m+10: LET n=0
4030 GO SUB 40
```

Ahora puede examinarse la subrutina de la línea 40 con detalle. En la página se obtiene fácilmente un cursor intermitente. Las funciones SCREEN$ y FLASH pueden combinarse para leer el carácter en un punto de la pantalla e imprimir en un cursor intermitente. La línea 40 genera el cursor y entonces el control pasa a la primera subrutina para leer un carácter del teclado. Dicha rutina es llamada para comenzar en la línea 11 en lugar de en la línea 10, de este modo las teclas que controlan el cursor pueden autorrepetirse. Una vez que se ha introducido el carácter, es necesario re-escribir el carácter sin FLASH antes de mover el cursor a la siguiente posición.

```
40 LET c$=SCREEN$ (m,n): PRINT
FLASH 1;AT m,n;c$
41 GO SUB 11: PRINT AT m,n;c$
```

Las líneas comprendidas entre la 42 y la 54 ajustan los valores de las variables 'm' y 'n' que indican las coordenadas en la pantalla. Las líneas comprendidas entre la 46 y la 49 se aseguran de que las coordenadas están realmente en el rango de la pantalla y la última línea envía el control al menú principal.

```
  42 IF b$="5" THEN  LET n=n-1:
LET c=c-1
  43 IF b$="6" THEN  LET m=m+1:
LET c=c+32
  44 IF b$="7" THEN  LET m=m-1:
LET c=c-32
  45 IF b$="8" THEN  LET n=n+1:
LET c=c+1
  46 IF n<0 THEN  LET n=n+32: LE
T m=m-1
  47 IF n>31 THEN  LET n=n-32: L
ET m=m+1
  48 IF m<0 THEN  LET c=c+32: LE
T m=m+1: GO TO 48
  49 IF m>19 THEN  LET m=m-1: LE
T c=c-32: GO TO 49
  50 RETURN
```

La computadora deberá responder a otras teclas además de aquellas que tienen flechas direccionales y que van desde el 5 al 8 en la línea superior. La tecla 'P' hará que el procesador avance una página en el texto y la tecla 'Q' le hará retroceder otra página. Cuando el cursor esté en el lugar deseado, la página seleccionada

se imprimirá con el cursor al principio de la línea 1Ø, apuntando el último carácter tratado. En este momento, puede hacerse que la computadora imprima el texto desde la posición del cursor o que acepte entrada de información desde el mismo lugar. Las teclas con las que se controlan estas operaciones son la 'Z' para introducir más texto y la 'C' (tecla LPRINT) para la impresión. Las últimas del siguiente bloque se aseguran de que el puntero tiene un valor adecuado antes de devolver el control al bucle.

```
4Ø5Ø IF b$="p" THEN  LET c=c+6Ø8
: GO TO 4ØØØ
4Ø6Ø IF b$="q" THEN  LET c=c-6Ø8
: GO TO 4ØØØ
4Ø7Ø IF b$="e" THEN  GO TO 42ØØ
4Ø8Ø IF b$="z" THEN  GO TO 412Ø
4Ø9Ø IF b$="c" THEN  LET x=c: CL
S : GO TO 3ØØ5
41ØØ IF m>19 OR m<Ø THEN  LET m=
Ø: LET n=Ø: GO TO 4ØØØ
411Ø GO TO 4Ø3Ø
```

Las cinco líneas siguientes son llamadas por la línea **4Ø8Ø** si la tecla 'Z' se presiona. Estas líneas preparan la impresión de tal forma que la siguiente posición esté en la izquierda de la línea inferior. Para situar el nuevo material dentro del contexto, se escribe el texto hasta la posición del cursor para juntarse con la nueva escritura. La línea **414Ø** localiza las coordenadas de la pantalla en las cuales se debe imprimir las partes de las páginas para que queden apropiadamente alineadas.

```
412Ø LET b=1: IF c<321 THEN  LET
 c=321
413Ø IF c>LEN a$ THEN  LET c=LEN
 a$
414Ø IF c<673 THEN  LET m=2Ø-INT
 (c/32): LET n=32-(c-INT (c/32)*
32): IF n=32 THEN  LET m=m+1: LE
T n=1
415Ø IF c<673 THEN  PRINT AT m,n
;a$( TO c-1): GO TO 1ØØØ
416Ø CLS : PRINT a$(c-672 TO c-1
): GO TO 1ØØØ
```

Si se hubiera pulsado la tecla 'E' entonces la línea **4Ø7Ø** hubiera enviado el control a este bloque que vamos a analizar. Estas

son las líneas más importantes de las rutinas de edición; permiten que se añada nuevo texto borrando de paso el anterior. Ambas operaciones se realizan rápidamente. La línea 4200 solicita instrucciones y envía después a la computadora a la siguiente sección si fuera necesario. La línea 4210 inicia el proceso de borrado subrayando el primer carácter a desaparecer. La subrutina de la línea 40 se utiliza para situar el cursor en el último carácter que debe borrarse; cuando se pulsa 'R' todos los caracteres entre las dos marcas desaparecen. La línea 4230 elimina parte del buffer del texto y coloca el puntero antes de retornar al bucle principal de edición.

```
4200 INPUT "D delete A add";b$:
IF b$<>"d" THEN  GO TO 4300
4210 LET g=c: PRINT  OVER 1;AT m
,n;"_"
4220 GO SUB 40: IF b$<>"d" THEN
 GO TO 4220
4230 LET a$=a$( TO g-1)+a$(c+1 T
O ): LET c=g-1: GO TO 4000
```

La última rutina de este bloque es incluso más corta que la anterior y utiliza la misma técnica de manipulación de cadenas.

```
4300 IF b$<>"a" AND b$<>"p" THEN
 GO TO 4120
4310 INPUT "type extra material
";b$: LET a$=a$( TO c)+b$+a$(c+1
 TO ): LET c=c+LEN b$: GO TO 400
0
```

La rutina que permite la grabación del texto es muy similar a aquella estudiada que permite la carga. La línea 5010 recuerda al usuario que tenga conectado un solo cable durante la grabación y después comienza a registrar en cinta a$.

```
5000 PRINT "Find a blank section
 of tape"'"Key a file ref. for t
he text": INPUT b$: PRINT b$''"K
ey C if correct": INPUT c$: CLS
: IF c$<>"c" THEN  GO TO 900
5010 PRINT " PULL OUT THE EAR PL
UG ": SAVE b$ DATA a$(): GO TO 9
00
```

La rutina de mantenimiento permite al usuario conectar y desconectar la impresora, así como, borrar todo el texto para introducir otro nuevo. La línea 6Ø1Ø inicializa las variables que son necesarias en el resto de las rutinas y si la bandera que indica la situación de la impresora 'p' vale 1, fija también el número de líneas por página. La línea 6Ø3Ø redimensiona el almacenamiento temporal del texto. Cuando el programa se carga por primera vez desde la cinta, no existe el espacio destinado al texto con objeto de minimizar el tiempo de carga. Cuando el programa se ha cargado debe pulsarse 'T'.

```
6ØØØ CLS : PRINT "Is the printer
 attached? (Y/N)": INPUT b$: IF
LEN b$<>1 OR (b$<>"y" AND B$<>"n
") THEN  GO TO 6ØØØ
6Ø1Ø LET x3=22: LET p=Ø: IF b$="
y" THEN  LET p=1: PRINT '"How ma
ny lines per page?": INPUT x3: P
RINT x3'"key C if correct": INPU
T b$: IF b$<>"c" THEN  CLS : GO
TO 6ØØØ
6Ø2Ø LET b=1: LET c=321: POKE 23
692,9
6Ø3Ø PRINT '"Key T to set up tex
t space.       CAUTION  destroys
existing text"
6Ø4Ø INPUT b$: IF b$="t" THEN  D
IM a$(84ØØ)
6Ø5Ø GO TO 9ØØ
```

La línea 61ØØ es la primera que debe ejecutarse cuando el programa se carga. Si se preciase utilizar en el texto dobles comillas, tendrían que ser falsas comillas, es decir, deberían obtenerse utilizando para ello uno de los caracteres definidos por el usuario. La línea 61ØØ envía el control al comienzo de la rutina de mantenimiento habiendo definido previamente unas falsas dobles comillas. La línea 7ØØØ es la última del programa y su función es exclusivamente señalar que el programa se ha acabado.

```
61ØØ FOR j=Ø TO 7: POKE USR "a"+
j,Ø: NEXT j: POKE USR "a"+1,36:
POKE USR "a"+2,36: GO TO 6ØØØ
7ØØØ PRINT AT 9,8;"END OF PROGRA
M"
```

El programa está casi listo para ser usado. Convendría hacer una copia del mismo. Pulse CLEAR (en la tecla 'X') y después ENTER. Registre el programa en cinta escribiendo 'SAVE «specpro» LINE 6100' y pulse ENTER. De este modo el programa se ejecutará automáticamente a partir de la línea 6100 cada vez que se cargue, no siendo preciso presionar RUN y después ENTER.

Variables de Specpro

b —posición dentro de la línea
c —número del carácter dentro del buffer del texto
d —número de línea
f —bandera para diversas rutinas
g —primer carácter a borrar en la rutina de edición
p —bandera para la impresora
m —número de línea del cursor
n —número de columna del cursor
x —primer carácter a imprimir
x1 —contador de líneas de la pantalla
x2 —número de página
x3 —líneas por página
x4 —líneas ya impresas en la página en curso
y —longitud de la línea, debe ser inferior a 34 caracteres
z —longitud de página
a$ —cadena de almacenamiento temporal del texto (buffer), dimensionado para 8400 caracteres
b$ —almacenamiento temporal
c$ —almacenamiento temporal

3 INSTRUCCIONES OPERATIVAS DEL SPECPRO

En la figura 3.1 se da un organigrama funcional de SPECPRO. Carguése desde la cinta escribiendo LOAD “ ” y después ENTER. El programa se autoejecuta nada más cargarse, comenzando por hacer algunas preguntas. Contéstese a la computadora que

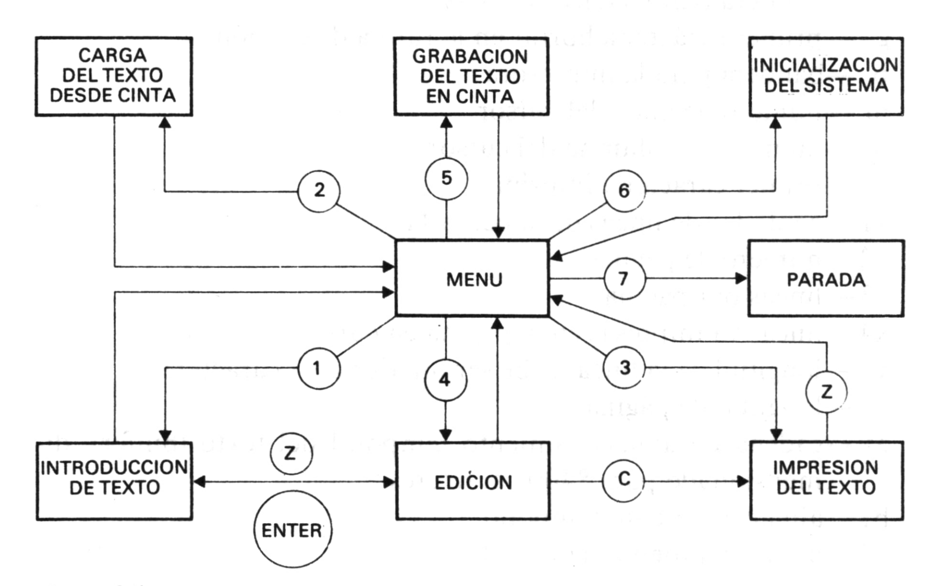


Figura 3.1

la impresora no está conectada aunque lo esté. La señal para indicar que la impresora está desconectada es pulsar ‘N’ y después ENTER. Es muy fácil volver a esta sección y decir a la computadora que la impresora está conectada cuando llegue el momento. En este instante es imprescindible preservar espacio para el almacenamiento del texto, por ello pulse ‘T’ seguido de ENTER cuando se le pregunte. Al utilizar el procesador de textos no es preciso presionar la tecla SHIFT para contestar a las preguntas que formula la computadora de vez en cuando. Todo está preparado para que se acepten las señales producidas por las letras minúsculas. En los recordatorios de la pantalla se han utilizado letras mayúsculas porque son más fáciles de leer.

La imagen que aparece en pantalla en este momento se denomina menú de opciones. Se elige presionando la tecla del número correspondiente a la opción seleccionada. El resto de las instrucciones operativas muestran la manera de utilizar las diversas opciones. En la figura 3.2 se ofrece una lista de las teclas con funciones especiales.

Opción 1 — escritura de texto

ENTER	posibilita la entrada de marcadores para un correcto formateo; véanse las indicaciones en pantalla ENTER mueve el control a la opción 4 "Z" da el menú
SHIFT/Ø	borra el último carácter

Opción 2 — carga del texto desde cinta

siga las instrucciones que aparecen en pantalla

Opción 3 — impresión del texto

"Z"	se abandona el menú

Opción 4 — edición del texto

"P"	se avanza una página
"Q"	se retrocede una página
5 a 8	control del cursor
"E"	edición desde la posición del cursor "A" insertar después del cursor (siga las instrucciones que aparecen en la pantalla) "D" borra — mueva el cursor a la última posición del texto que deba borrarse y pulse otra vez "D"
"Z"	para escribir a partir de la posición del cursor
"C"	para imprimir en papel a partir de la posición del cursor

Opción 5 — grabación del texto en cinta

siga las instrucciones que aparecen en pantalla

Opción 6 — mantenimiento

siga las instrucciones que aparecen en pantalla

Figura 3.2

ENTRADA DE TEXTO. OPCION 1

Pulse la tecla '1' y la pantalla se borrará excepto una pequeña línea en la parte inferior. Así se indica el lugar donde aparecerá el

primer carácter que se pulse. Pruebe introduciendo un texto, si se equivoca pulse (SHIFT/Ø) y podrá borrarlo apareciendo además un cursor temporal que permite conocer la posición en la línea. Conforme se van escribiendo líneas, éstas se van corriendo hacia posiciones superiores en la pantalla, llega un momento en que las líneas van desapareciendo por la parte superior de la pantalla. Sin embargo, el texto no se ha perdido, sino que continua almacenado en la computadora y es muy sencillo re-escribirlo en la pantalla.

No es necesario ser especialmente cuidadoso con la escritura. De hecho es preferible no intentar que las palabras acaben al final de la línea dejando espacios extra. El programa organiza el texto de tal forma que ninguna palabra quede partida entre dos líneas, cada línea se corta en la última palabra completa y se añaden los espacios necesarios para su correcta justificación. Si el usuario intenta sustituir a la computadora en esta función todo irá peor. La copia impresa será más vistosa si se permite a la computadora que organice el texto por sí sola. El programa ha sido escrito para hacer el mejor uso posible del espacio disponible. No es necesario dar espacios cada vez que se empieza una nueva línea, párrafo o página, los marcadores especiales añadidos al texto indican a la computadora cuando comienza una nueva línea, párrafo o página.

Presione ENTER cuando desee introducir algún marcador. Entonces aparecerá una línea de recordatorios en la parte inferior de la pantalla. Pulse la tecla del marcador que desee y se añadirá al texto. No se necesita nada más, por tanto, no es preciso añadir espacios extra ni tampoco forzar una justificación manual que la computadora hace automáticamente. La computadora cuenta las líneas que imprime en papel y escribirá un número de página cuando ésta se ha acabado. Si se pide que acabe una página antes, lo que en circunstancias normales correspondería, el resto de la página se rellena con líneas en blanco y al final se añade el número de página.

Hay dos teclas especiales que pueden pulsarse en este momento, 'Z' y ENTER. La tecla 'Z' se utiliza como señal de escape y permite volver al menú. ENTER da acceso a las rutinas de edición. A estas rutinas también se puede acceder desde el menú, pero es conveniente que el cambio entre introducción y edición sea lo más simple posible. Si, por ejemplo, se observa una palabra de menos en una frase anterior, se puede pasar a la rutina de edición, hacer la corrección y volver a la introducción de texto.

CARGA DE TEXTO DESDE CINTA. OPCION 2

Pulsando la opción 2 del menú, la máquina se prepara para cargar texto previamente registrado en cinta. Se necesita saber en qué parte de la cinta ha sido grabado y el nombre que se le dio. Siga las instrucciones que aparecen en pantalla y conteste las preguntas. Si comete algún error pulse ENTER cuando se le pregunte si la información es correcta y retornará al menú. Asumiendo que todo está bien cuando el texto se ha cargado aparecerá el menú. Si resulta que no es posible cargar el texto, lo que podría suceder si se hubiera grabado otra cosa encima de parte del texto, puede abandonarse la rutina de carga presionando SHIFT y BREAK. El programa puede re-emprenderse desde el menú pulsando GO TO 1 y después ENTER. No pulse RUN o perderá el texto.

IMPRESION DEL TEXTO. OPCION 3

Para imprimir todo el texto pulse la opción 3 del menú. Si se ha advertido a la computadora que la impresora está conectada, imprimirá el texto en papel después de que haya sido preparado en la pantalla. Puede engañarse a la computadora diciéndola que no hay impresora y no intentará escribir en papel. De hecho, la impresora se conecta de este modo, visitando para ello la rutina de mantenimiento-opción 6. Si desea comenzar la impresión desde otro carácter distinto del primero consulte la sección dedicada a la rutina de edición-opción 4. Para volver al menú antes de finalizar el proceso de impresión pulse 'Z'.

EDICION. OPCION 4

Para manejar esta parte del programa se necesitan algunas teclas más. En primer lugar, si se ha accedido a esta rutina por error pulse 'Z' y volverá al modo de introducción de texto, donde podrá seguir escribiendo desde la posición intermitente del cursor. Suponiendo que se haya accedido a la rutina de edición deliberadamente, ¿cómo modificar el texto? Lo primero que hay que hacer es mover el cursor hacia el lugar donde se desea hacer la corrección. Si se desea añadir material, el cursor debe estar en el lugar inmediatamente anterior a donde se quiere insertar el texto. Si se desea borrar texto, posicionar el cursor sobre el primer carácter a borrar. Si se desea escribir encima del antiguo texto desde un punto determinado, el cursor debe ponerse en el lugar desde el que se quiere seguir escribiendo.

Para mover el cursor se tienen que utilizar las teclas con flechas (desde el 5 al 8). Si el texto requerido no está en la página que aparece en pantalla pulsar 'P' para avanzar o 'Q' para retroceder una página. Cuando el cursor está en el lugar adecuado hay tres teclas que se pueden pulsar. 'Z' devuelve el control al modo de introducción de texto. 'C' es la tecla LPRINT y permite comenzar la impresión del texto desde el lugar señalado por el cursor y por último 'E' permite añadir o eliminar caracteres. Para añadir caracteres pulse 'A' y ENTER entonces se le pedirá que introduzca lo que desee. La inserción se llevará a cabo cuando pulse ENTER. Si pulsa la tecla 'D' para borrar, el carácter será subrayado. Utilice las teclas de control del cursor para situarle sobre el último carácter que va a borrarse; pulsando otra vez 'D' el texto desaparecerá.

Al finalizar estas rutinas de edición el texto se centrará en la pantalla de modo que el punto donde tuvo lugar el cambio quede en el centro. Puede moverse el cursor al lugar que se desee y pulsar 'Z' con lo que se podrá escribir sobre el texto anterior. Recuerde que con 'P' y 'Q' se avanza o retrocede una página cada vez. Si quiere volver al menú, pulse 'Z', después ENTER y luego 'Z' de nuevo. Parece más complicado de lo que es en realidad.

GRABACION DE TEXTO. OPCION 5

Esta rutina es similar a la que permite la carga de texto desde cinta. Siga las instrucciones que aparecen en pantalla. Puede retornar al menú pulsando ENTER cuando le pidan que pulse 'C' para confirmar el nombre dado al texto.

MANTENIMIENTO. OPCION 6

Con esta rutina se puede conectar y desconectar la impresora. Si se dice a la computadora que hay impresora, preguntará cuantas líneas debe imprimir antes del número de página y comenzar una página nueva. Alrededor de 75 líneas en papel tamaño A4 es una medida razonable. Si va a comenzar un texto nuevo tendrá que pulsar 'T' para borrar el texto anterior. Desde esta rutina el control vuelve al menú principal.

PARADA. OPCION 7

Esta opción es solo una manera elegante de detener la ejecución de un programa. La rutina devuelve el control al BASIC.

4 SPECTEXT

El programa que se ha desarrollado hasta el momento permite introducir y editar textos. El texto puede almacenarse en cinta y, naturalmente, volver a cargarse para re-editar, imprimir, etc., tantas veces como se desee. Los procesadores de textos que se comercializan ofrecen más facilidades que este programa. No obstante, hay que tener en cuenta que los procesadores de textos que se comercializan son frecuentemente del mismo precio que el computador con el que estamos trabajando. Por tanto debemos esperar mucho más de un gasto de esta cuantía. Lo mejor es decidirse por alguno de los programas escritos para el Timex Sinclair TS2048 que son tan buenos como los de otras computadoras similares.

Si algunos programas rivalizan con el software que se comercializa, cabe preguntarse ¿con qué problemas nos enfrentaremos? En primer lugar, se necesita más capacidad de almacenamiento. 8500 caracteres corresponden aproximadamente a unas 1500 palabras inglesas, suficientes para la mayor parte de las cartas pero no para artículos de prensa. Un artículo medio tiene alrededor de 2000 palabras. Esta dimensión añade dificultades adicionales que exigen más del procesador de textos. Por ejemplo, sería útil poder marcar una parte del texto para trasladarla a otro lugar del mismo. La posibilidad de encontrar en el texto todas las ocurrencias de una palabra o frase dadas y sustituirlas después por otra palabra o frase, sería, asimismo, muy útil. Si estas características son útiles deben poderse realizar. Ahora bien, el procesador de textos debe estar diseñado para ahorrar tiempo, por tanto, todas estas facilidades deben realizarse casi instantáneamente. Es perfectamente posible añadir estas facilidades al procesador de textos en BASIC, pero el tiempo que utilizaría la computadora nos obligaría a volver a la máquina de escribir. El BASIC es demasiado lento para este tipo de trabajos. La respuesta obvia a este problema es el lenguaje máquina, pero los lenguajes de bajo nivel nos asustan a todos. El problema no es tanto escribir el programa en código máquina como el probarlo y corregirlo después. No existen mensajes de error que nos ayuden a adivinar donde nos habíamos equivocado. Los errores de un programa escrito en código máquina en el mejor de los casos pueden alterar el propio código

y en el peor de los casos pueden obligarnos a tener que desconectar la computadora perdiendo todo el programa. Los programas en código máquina deben grabarse en cinta antes de probarse, de tal forma que puedan ser cargados si la computadora se para durante el proceso de comprobación. Cualquier nueva versión del programa debe registrarse en cinta.

El método desarrollado para crear programas en lenguaje máquina para este procesador de textos avanzado es menos peligroso y más rápido y productivo. Por medio de esta nueva metodología, los programas en código máquina pueden escribirse utilizando las facilidades que ofrece el BASIC. La metodología y todas las rutinas que se han desarrollado utilizándola se explican con detalle en el Capítulo 6. En este capítulo se estudia la parte del procesador escrita en BASIC. Gran parte del programa se encarga de inicializar y llamar a las rutinas escritas en lenguaje máquina.

Una de las cosas que no deben perderse es la simplicidad de SPECPRO. No obstante, debe introducirse algunas teclas de control nuevas y debe ayudarse al usuario con nuevos recordatorios escritos en la pantalla. Los procesadores de textos comerciales son caros porque necesitan un manual de operaciones grueso para cubrir todas sus facilidades y características. SPECTEXT puede manejarse a través de todas las facilidades adicionales por medio de las dos teclas que se utilizaban en SPECPRO-'Z' y ENTER. Esto ha sido posible porque se ha desarrollado una nueva forma de presentar las opciones al usuario que se explicará más adelante.

Si se quiere mejorar la utilización de la memoria disponible, debe encontrarse otro método para añadir y suprimir texto al buffer. La manipulación de cadenas es rápida y flexible pero es la razón principal por la que el almacenamiento en el buffer es tan limitado en el primer programa. Para editar con la rapidez que se precisa y sin la manipulación de cadenas se necesita del código máquina, para ello se ha escrito la rutina MOVE. MOVE hace exactamente lo que indica su nombre; mueve bloques de texto de un lugar a otro de la memoria, miles de bytes en una fracción de segundo. En el capítulo destinado al código máquina hay un conjunto de diagramas que muestran cómo se lleva a cabo la edición por este medio. La gran ventaja de este método es que no se tiene que realizar una copia del buffer del texto. Se necesita poco más que el espacio ocupado por el texto, por lo tanto el buffer puede ocupar la mayor parte del espacio disponible.

La otra ventaja es que de este modo es más rápida la búsqueda de material en el buffer. La rutina FIND es un poco más larga que MOVE, pero es igualmente rápida, siendo capaz de examinar 25 000 caracteres en menos de un segundo. Se necesita otra rutina

capaz de rellenar un área de memoria con un carácter seleccionado. Esta rutina se llama FILL y se necesita para limpiar parte de la pantalla. Es un poco más complicada que MOVE. Con este pequeño grupo de rutinas rápidas pueden implementarse todas las facilidades que se echaban de menos en SPECPRO. La búsqueda y sustitución utilizan la rutina FIND para encontrar los caracteres especificados, MOVE para dejar espacio disponible para la sustitución y después MOVE de nuevo para insertar el nuevo material en el espacio preparado al efecto. La reubicación de parte del texto se realiza moviendo los caracteres al espacio suplementario y utilizando después MOVE para eliminar el espacio inútil y crear otro idéntico después en el nuevo lugar. La misma rutina

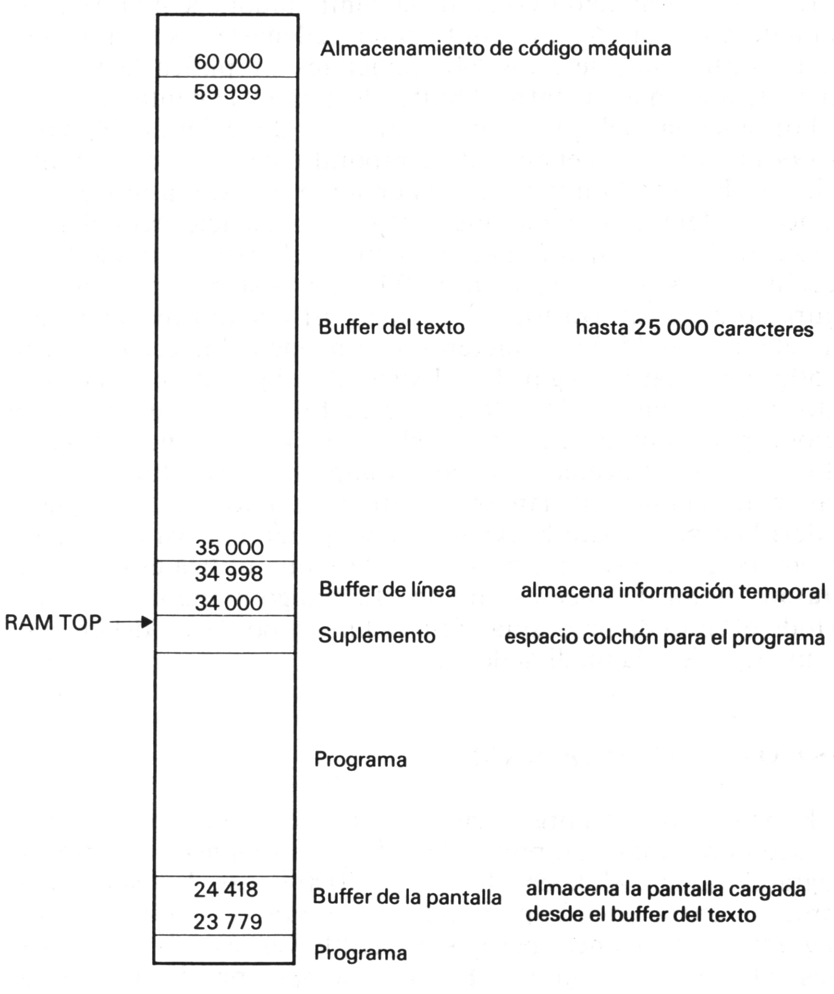


Figura 4.1

mueve después los caracteres desde el almacenamiento temporal al nuevo espacio.

Es útil conocer como se asigna la memoria a las distintas partes del programa. La figura 4.1 es el mapa de memoria de la computadora cuando se instala SPECTEXT. La primera de las cuatro áreas principales contiene el programa BASIC y hay una sección importante dentro de ella. El área comprendida entre las direcciones 23 779 y 24 418 actúa como almacenamiento temporal de la pantalla y a ella va a parar mediante MOVE cualquier material cuyo destino sea ser impreso. Esta área comprende 640 espacios comprendidos entre dobles comillas para delimitar el principio y fin de una cadena. Esta extraña idea se comprenderá mejor cuando se den los detalles del programa más adelante. El fichero de representación visual de la computadora se manipula de un modo muy complejo y resulta difícil rellenarle desde la memoria. La utilización de estos 640 caracteres simplifica la transferencia de información entre el buffer del texto y la pantalla.

Por encima del programa, entre las direcciones 34 000 y 34 999 hay un almacenamiento temporal conocido como 'buffer de línea'. Es aquí donde las líneas de texto que van a moverse se almacenan temporalmente mientras se crea espacio para ellas en otra parte. El principal almacenamiento del texto va desde la dirección 35 000 a la dirección 59 999, no obstante, para mayor seguridad no se utiliza todo el espacio. El tamaño recomendado para el buffer es 15 000 caracteres pero pueden almacenarse hasta 24 500 con relativa seguridad. Existe el peligro de machacar el texto si se utilizan los 25 000 caracteres. En este capítulo se da un método para limitar el tamaño del buffer. Por encima del buffer del texto está almacenado el código máquina. Las rutinas en lenguaje máquina no rellenan gran parte de esta área, puesto que se ha dejado espacio para la expansión del programa, así como para un programa que se verá más adelante y que utiliza este espacio para un 'buffer de línea extendido'. El código máquina se utiliza en todo el resto de programas de este libro y por tanto deben ocupar un espacio a la medida de todos.

LISTADO DEL PROGRAMA

El programa se ha organizado de forma que se mejore su lectura haciéndole más comprensible. El menú organiza el resto del programa dividiéndole en bloques de 1000 líneas. Las subrutinas empiezan en la línea 2; podríamos comenzar la explicación del programa en esta línea, pero es preferible ver las subrutinas después del resto del programa. El menú de opciones de la línea 900

es corto pero está completamente protegido contra el usuario que intente averiguar lo que sucede si elige la opción 8 en un menú que sólo contiene 7 posibilidades. La subrutina de la línea (10) alimenta la variable b$ con el primer carácter introducido desde el teclado. A lo largo del libro las referencias a subrutinas se enfatizan encerrando el número de línea entre paréntesis. De cuando en cuando, se recordará al usuario este convenio. La línea 920 prepara la pantalla para la acción. Si se selecciona la primera opción (introducción de texto), se escribe una pequeña marca en la parte limpia de la pantalla para mostrar dónde se escribirá el primer carácter. La pantalla se limpia con cualquiera de las opciones del menú, por este motivo hay muchas instrucciones CLS en el resto del programa. Si se pulsa la tecla correspondiente a una de las opciones del menú, la línea 930 envía el control a la línea apropiada.

```
 900 CLS : PRINT TAB 11;" SPECTE
XT "''''"1   Enter text"'"2   Loa
d text from tape"'"3   Print tex
t"'"4   Read through/edit text"'
"5   Save text on tape"'"6   Re-
organise"'"7   Stop work"
 910 GO SUB 10: IF LEN b$<>1 OR
b$<"1" OR b$>"7" THEN GO TO 910
 920 CLS : IF b$="1" THEN  PRINT
 AT 21,0;"_";AT 21,0;
 930 GO TO VAL b$*1000
```

La primera de las opciones corresponde a la introducción de texto y será probablemente la que se vaya a utilizar con mayor frecuencia, por eso se ha colocado en la línea 1000. La rutina (10) que escudriña el teclado, se llamará una vez por cada carácter. Cada vez que se la llama se modificará el contador de caracteres 'c'. La línea 1000 llama a (10) y después comprueba que el nuevo carácter no empuja 'c' más allá de los límites marcados por el espacio disponible. La variable 'c2' contiene este límite. Existe una rutina en la línea 1500 que avisa mediante un tono audible de que 'c' ha alcanzado el límite. Si todo va bien la línea 1010 filtra los posibles caracteres problemáticos y la línea 1015 cambia las dobles comillas que se introducen. El método de representación del texto en la pantalla imposibilita el uso de las dobles comillas, por eso uno de los caracteres definidos por el usuario (número 114) se ha diseñado para utilizarse en lugar de las dobles comillas (CODE 34), los caracteres que sobrepasan esta fase son almacenados mediante POKE en el buffer del texto e impresos después en

la pantalla. La línea 1020 realiza este trabajo y actualiza después las variables. La variable 'b' almacena la posición en la línea de la pantalla y cuando la última línea se ha completado, la computadora preguntaría si realiza el SCROLL. Almacenando mediante POKE el valor 9 en la dirección 23 692 el texto se correrá automáticamente hacia arriba sin necesidad de que la computadora se detenga pidiendo instrucciones. La variable 'c1' muestra hasta donde ha llegado el usuario dentro del buffer. Las rutinas de edición mueven 'c' dentro del buffer pero sólo entre 35 000 y el valor de 'c1'. La línea 1025 actualiza 'c1' y después la línea 1030 envía la computadora en busca de otro carácter.

```
1000 GO SUB 10: IF c>c2 THEN  GO
 TO 1500
1010 IF CODE b$<32 OR CODE b$>12
7 THEN  GO TO 1040
1015 IF CODE b$=34 THEN  LET b$=
CHR$ 144
1020 POKE 34999+c,CODE b$: PRINT
 b$;: LET b=b+1: LET c=c+1: IF b
=33 THEN  LET b=1: POKE 23692,9
1025 IF c>c1 THEN  LET c1=c
1030 GO TO 1000
```

La computadora sólo alcanza la línea 1040 si el carácter necesita un procesamiento especial. Para la mayor parte de ellos el 'tratamiento' consiste simplemente en rechazarlos. Sólo dos instrucciones necesitan ser interpretadas ENTER y DELETE. La línea 1040 inicializa la variable 'd' que contiene el número de línea dentro de la pantalla a 21 y la línea 1050 pasa el control a la siguiente sección si el carácter es distinto de DELETE. Las dos líneas siguientes eliminan el último carácter del buffer del texto y de la pantalla y arreglan las variables de modo que el siguiente carácter se acople adecuadamente. Finalmente, se lee un nuevo carácter por medio de (10) y el control vuelve al principio de la rutina.

```
1040 LET d=21
1050 IF CODE b$<>12 THEN  GO TO
1080
1060 LET b=b-1: POKE 34999+c,COD
E " ": LET c=c-1: IF b=0 THEN  L
ET b=32: LET d=d-1: PRINT " ";AT
 19,0;
```

```
1070 PRINT AT d,b-1;"_ ";AT d,b-
1;: GO SUB 10: GO TO 1050
```

La línea 1080 acepta los caracteres cuyos códigos están comprendidos entre 32 y 127, todos los demás excepto ENTER son rechazados por la línea siguiente. El rechazo se produce mandando el control a la línea 1000 para solicitar un nuevo carácter.

```
1080 IF CODE b$<128 AND CODE b$>
31 THEN  GO TO 1010
1090 IF CODE b$<>13 THEN  GO TO
1000
```

La tecla ENTER proporciona varias posibilidades y el recordatorio de la línea 1100 ayuda al usuario. Una serie de marcadores insertados en el texto indican a la computadora dónde empieza una nueva línea, párrafo o página; dichos marcadores deben introducirse inmediatamente después que ENTER. La tecla 'Z' permite al usuario retornar al menú mientras que pulsar de nuevo ENTER implica pasar el control a la rutina de edición. La línea 1160 añade el carácter marcador al texto y a la pantalla antes de que la máquina retorne al principio de la rutina.

```
1100 INPUT "1line 2para 3page (Z&
ENTER)";b$: IF LEN b$<>1 OR (b$<
>"z" AND (b$<"1" OR b$>"3")) THE
N  GO TO 4000
1110 IF b$="z" THEN  GO TO 1
1120 IF b$="1" THEN  LET b$=CHR$
 131
1130 IF b$="2" THEN  LET b$=CHR$
 139
1140 IF b$="3" THEN  LET b$=CHR$
 134
1150 POKE 34999+c,CODE b$: PRINT
 b$;: LET c=c+1: LET b=b+1: IF b
>31 THEN  LET b=1
1160 GO TO 1000
```

La línea 1500 llama la atención sobre el fin del espacio disponible y envía el control al menú si el usuario pulsa la tecla 'Z'.

```
1500 BEEP .1,20: PRINT AT 9,9;"
BUFFER FULL  ": IF INKEY$<>"z" T
```

```
HEN  GO TO 15ØØ
151Ø GO TO 1
```

La segunda opción carga el texto desde la cinta y es sencilla en comparación con la primera. La línea 2020 inicializa ciertas variables para que la máquina empiece la carga adecuadamente y la línea 2030 efectúa realmente la carga. El texto se almacena como bytes y por tanto la instrucción de carga debe ser LOAD-CODE. El texto se registra con dos bytes extras al principio. Estos bytes han sido cargados con el valor de 'c1' mediante la instrucción POKE, justo antes de la grabación del texto; de este modo, la línea 2040 es capaz de inicializar esta variable para la nueva ejecución.

```
2ØØØ PRINT "Key the name of the
text": INPUT b$: PRINT 'b$
2Ø1Ø PRINT '"key C to confirm":
INPUT c$: IF c$<>"c" THEN  CLS :
 GO TO 1
2Ø2Ø LET p=Ø: LET b=1: LET c=b:
POKE 23692,9
2Ø3Ø PRINT '"play tape": LOAD b$
CODE
2Ø4Ø LET c1=PEEK 34998+256*PEEK
34999
2Ø5Ø GO TO 1
```

La tercera opción es mucho más complicada que las dos anteriores, pero es prácticamente idéntica a la rutina de formateo del procesador de textos en BASIC. En el Capítulo 2, páginas 11-15, se encontrará documentación suficiente sobre esta sección. A continuación sigue la versión de esta rutina que realiza el formateo en este programa.

```
3ØØØ LET x=te
3Ø1Ø LET x1=1: LET x2=x1: LET x4
=x1
3Ø2Ø LET y=31
3Ø3Ø LET f=PEEK x: IF f=144 THEN
  GO TO 3Ø5Ø
3Ø4Ø IF f>127 THEN  LET x=x+1: I
F f<>131 THEN  GO TO 32ØØ
3Ø5Ø IF INKEY$="z" THEN  GO TO 1
3Ø55 IF x+y-te>c1 THEN  PAUSE 5Ø
Ø GO TO 1
```

```
3Ø6Ø LET f=Ø: LET b$="": FOR j=x
 TO x+y: LET b$=b$+CHR$ PEEK j:
IF (PEEK j>127 AND PEEK j<>144)
AND f=Ø THEN  LET f=j
3Ø7Ø NEXT j: LET y=j: IF f>Ø THE
N  LET y=f
3Ø8Ø LET f=Ø: FOR j=x TO y: IF P
EEK j=32 THEN  LET f=1: LET j=y
3Ø9Ø NEXT j: IF f=Ø THEN  GO TO
311Ø
31ØØ IF PEEK y<>32 AND PEEK y<12
8 THEN  LET y=y-1: GO TO 31ØØ
311Ø GO SUB 2Ø: IF PEEK y>127 TH
EN  LET x=x-1
312Ø GO SUB 34ØØ: GO TO 3Ø2Ø
32ØØ IF f<>139 THEN  GO TO 33ØØ
321Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: LET a=x3-x4: IF a<3 THEN  F
OR j=1 TO a: PRINT : GO SUB 21:
GO SUB 34ØØ: NEXT j: GO TO 323Ø
322Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: PRINT : GO SUB 21: GO SUB 3
4ØØ
323Ø PRINT TAB 5;: LET y=26: GO
TO 3Ø3Ø
33ØØ IF f<>134 THEN  LET x=x+1:
GO TO 3Ø2Ø
331Ø LET a=x3-x4: PRINT : GO SUB
 21: FOR j=1 TO a: PRINT : GO SU
B 21: NEXT j: GO SUB 341Ø
332Ø GO TO 3Ø2Ø
34ØØ IF x4<x3 OR p=Ø THEN  RETUR
N
341Ø FOR j=1 TO 5: PRINT : GO SU
B 21: IF j=3 THEN  PRINT TAB 15;
x2: LET x2=x2+1: GO SUB 21
342Ø NEXT j: LET x4=1: RETURN
```

La cuarta opción se refiere a la edición del texto y está basada en la misma opción del procesador de textos en BASIC. Sin embargo, contiene tantas facilidades adicionales que se precisa nueva documentación. Es la parte más larga del programa con mucha diferencia sobre las demás y contiene sus propios menús. Uno de estos submenús dispone de seis opciones, por tanto a nadie debe

sorprender que la rutina de edición ocupe alrededor de una tercera parte del programa.

En las dos primeras líneas se asegura de que el contador de caracteres se encuentre dentro de los límites del buffer del texto. La rutina (95) localiza la sección del texto que contiene el carácter al que apunta 'c'. El texto se imprime de manera que este carácter aparezca en el borde izquierdo y hacia la mitad de la pantalla. La rutina (40) se encarga del control del cursor y permite que se desplace desde su lugar original hacia cualquier otra posición en la pantalla. En este momento el programa responde a varias teclas además de aquellas señaladas con flechas y que se utilizan normalmente para desplazar el cursor. Las líneas comprendidas entre la 4050 y la 4090 ejecutan una acción determinada, dependiendo de la tecla pulsada 'P', 'Q', 'E', 'Z', o 'C'. Mediante 'P' se desplaza el cursor una página hacia adelante y mediante 'Q' se retrocede una página. La inserción y borrado de partes del texto se realizan pulsando 'E' (editar) y con la tecla 'C' se ejecuta la rutina de formateo e impresión desde la posición del cursor. 'C' es la tecla LPRINT.

Con la tecla 'Z' se abren otras seis opciones que aparecerán en la pantalla mediante un submenú especial. Las coordenadas en la pantalla del cursor son 'm' y 'n'; la línea 4100 se encarga de que permanezcan dentro de los límites de la pantalla antes de que la línea 4110 envíe el control de nuevo a la subrutina (40).

```
4ØØØ IF c<1 THEN  LET c=1
4Ø1Ø IF c>c1 THEN  LET c=c1
4Ø2Ø GO SUB 95
4Ø3Ø GO SUB 4Ø
4Ø5Ø IF b$="p" THEN  LET c=c+6Ø8
: GO TO 4ØØØ
4Ø6Ø IF b$="q" THEN  LET c=c-6Ø8
: GO TO 4ØØØ
4Ø7Ø IF b$="e" THEN  GO TO 42ØØ
4Ø8Ø IF b$="z" THEN  GO TO 46ØØ
4Ø9Ø IF b$="c" THEN  LET x=c+349
99: CLS : GO TO 3ØØ5
41ØØ IF m>19 OR m<Ø THEN  LET m=
Ø: LET n=Ø
411Ø GO TO 4Ø3Ø
```

Cuando se pulsa la tecla 'E' en la última rutina, se está indicando que el usuario desea insertar o borrar caracteres; en la línea 4200 se solicita aclaración sobre este particular. Es preciso ajustar

'c' para mejorar el aspecto en pantalla de la primera parte del texto. En la línea 4205 se filtran las situaciones más difíciles y la inserción para su tratamiento posterior; en este momento la línea 4210 comienza el proceso de borrado subrayando el primer carácter destinado a desaparecer y almacenando su posición en el buffer en 'g'. La rutina (40) que controla el cursor entra de nuevo en juego para que pueda desplazársele hacia un nuevo lugar. La línea 4220 continúa llamando a la rutina (40) hasta que se pulsa la tecla 'D' por segunda vez para señalar el final del párrafo que debe desaparecer.

```
4200 INPUT "D delete A add";b$:
IF c<321 THEN  LET c=c-1
4205 IF b$<>"d" OR c<1 OR c>c1 T
HEN  GO TO 4300
4210 LET g=c: PRINT  OVER 1;AT m
,n;"_"
4220 GO SUB 40: IF b$<>"d" THEN
 GO TO 4220
```

En algunas ocasiones, particularmente cuando 'c' es menor que 'g', el borrado podría ocasionar problemas, la línea 4225 se ocupa de evitarlos. El material borrado no se destruye de inmediato, sino que, se almacena en el área de 1K conocida como buffer de línea y situada en la dirección 34 000. Se utiliza el carácter CHR$ 255 para etiquetar el buffer de línea indicando que contiene información borrada. Hasta que no se escribe de nuevo sobre esta etiqueta, el texto borrado puede reescribirse en el mismo u otro lugar, utilizando para ello la opción de mover que se comenta más tarde. La primera parte de la línea 4230 prepara esta tarea y la segunda parte mueve el resto del texto hacia abajo para «machacar» el material que ha de borrarse. La variable 'v' alimenta la rutina MOVE escrita en lenguaje máquina a la que se recurre desde la subrutina (30). En la línea 4240 se concluye el trabajo sustrayendo de los punteros 'c' y 'c1' del buffer del texto el número de caracteres que tienen que desaparecer.

```
4225 IF c<g OR c<1 OR c>c1 THEN
 GO TO 4000
4230 LET b=c-g+1: LET v1=b: LET
v2=34003: LET v3=te+g: GO SUB 30
: RANDOMIZE b: POKE li,255: POKE
 34001,PEEK q: POKE 34002, PEEK r
: LET v1=c1-c: LET c=c+1: LET v2
```

```
=te+g: LET v3=te+c: GO SUB 3Ø
424Ø LET a=c-g: LET c=c-a: LET c
1=c1-a: GO TO 4ØØØ
```

Antes de que pueda insertarse texto, la línea 4300 elimina los problemas trasladándolos a la línea 4000. Afortunadamente esta línea es capaz de solventarlos. Las inserciones precisan una llamada a la subrutina (60), donde se procesa el texto suplementario. La variable 'li' contiene la dirección del principio del área destinada al buffer de línea, 34 000. Este número y 35 000 son utilizados a menudo en el programa, por lo cual se ahorra bastante memoria almacenando sus valores en dos variables, 'li' y 'te' que se utilizan donde es necesario. La línea 60 crea otra variable 'c3' para indicar el último lugar en el buffer de línea que está ocupado por el nuevo texto introducido. Si 'c3' menos 'li' no es mayor o igual que uno, entonces no hay texto que insertar y lo más fácil es volver al principio de la rutina de edición. Esto se comprueba en la línea 4320. El resto de esta sección prepara las cosas para que las rutinas en lenguaje máquina realicen los movimientos necesarios y después de haber llamado a las rutinas (30) y (35) incrementa el valor de 'c' y 'c1'. Los diagramas detallados de los movimientos de datos se encuentran en el Capítulo 6, páginas 78 y 79.

```
43ØØ IF b$<>"a" OR c<1 OR c>=c1
THEN  GO TO 4ØØØ
431Ø LET c=c+1: GO SUB 6Ø
432Ø LET a=c3-li: IF a<1 THEN  G
O TO 4ØØØ
433Ø LET v1=c1-c+1: LET v2=te+c1
+a: LET v3=te+c1: GO SUB 35
434Ø LET v1=a: LET v2=te+c: LET
v3=li: GO SUB 3Ø
435Ø LET c1=c1+a: LET c=c+a: GO
TO 4ØØØ
```

Si se pulsa la tecla 'Z' durante la secuencia de control del cursor se dispondrá de otro conjunto de opciones para que elija el usuario. La elección es más fácil si el texto continúa en pantalla, por tanto cualquier menú que se utilice aquí no deberá interferir con el texto en la pantalla. Con las siguientes líneas se muestra en pantalla un menú, de manera que aparecerá una opción cada vez en la parte inferior de la pantalla. Pulsando ENTER se rechaza la opción en activo y se escribe la siguiente en la pantalla. Se elige una opción pulsando cualquier tecla distinta de ENTER mientras

que está en activo. La línea 46ØØ comprueba que 'c' está en el rango correcto y después se asegura que el cursor intermitente permanece durante la secuencia del menú antes de ofrecer la primera opción. Los nombres de las opciones se han elegido de manera que contengan el mismo número de caracteres, así el menú es más vistoso que si cada opción tuviera una longitud diferente.

```
46ØØ GO SUB 85: PRINT  FLASH 1;A
T m,n;SCREEN$ (m,n): INPUT "type
?";b$: IF b$="z" THEN  LET c=c+1
: GO TO 412Ø
46Ø1 INPUT "edit?";b$: IF b$="z"
 THEN : GO TO 4ØØØ
46Ø2 INPUT "seek?";b$: IF b$="z"
 THEN  GO TO 461Ø
46Ø3 INPUT "alter";b$: IF b$="z"
 THEN  GO TO 463Ø
46Ø4 INPUT "move?";b$: IF b$="z"
 THEN  GO TO 48ØØ
46Ø5 INPUT "menu?";b$: IF b$="z"
 THEN  GO TO 1
46Ø6 GO TO 46ØØ
```

La opción 'tipo' envía a la computadora a un conjunto de líneas que se dejaron antes para más tarde y que ahora entran en juego. Si 'c' está suficientemente en el interior del buffer todo resulta sencillo. La rutina (85) se ocupa de mantener a 'c' dentro del buffer y la rutina (1ØØ) limpia el texto viejo del almacenamiento de la pantalla y se actualizan los valores de las variables 'v', para cargar una nueva parte del texto desde el buffer. La rutina (3Ø) recurre al código máquina para realizar el movimiento y la impresión, después envía el control a la primera línea de la rutina de entrada del texto.

```
412Ø LET b=1
413Ø GO SUB 85: GO SUB 1ØØ: IF c
<64Ø THEN  GO TO 416Ø
414Ø LET v1=64Ø: LET v2=23779: L
ET v3=c+34359: GO SUB 3Ø
415Ø CLS : GO SUB 2: PRINT AT 1,
Ø;a$;: GO TO 1ØØØ
```

Si 'c' apunta hacia una posición inferior a la 640 dentro del buffer del texto, entonces las líneas 416Ø y 417Ø se aseguran de

que el texto se presente de tal modo que los nuevos caracteres se concatenen perfectamente con el texto anterior.

```
4160 LET c=c-1: IF c=1 THEN  CLS
 : PRINT AT 21,0;: GO TO 1000
4170 LET v1=c: LET v2=24418: LET
 v3=34999+c: GO SUB 35: LET c=c+
1: GO TO 4150
```

La opción de buscar obliga a entrar en juego al código máquina. La rutina (90) pregunta si la máquina debe comenzar la búsqueda desde la posición del cursor o desde el principio del texto y la rutina (80) obliga al código máquina a realizar su papel. Las rutinas en lenguaje máquina sitúan una bandera de 'localizado' y devuelven una dirección en el lugar 60 135 de la memoria, para indicar donde se encuentra el material buscado. Si no se ha colocado la bandera, esto implica que se ha llegado a 'c1' sin encontrar el material que se andaba buscando. Si el usuario ordena nuevas búsquedas del mismo material, la línea 4620 organiza la repetición de la búsqueda desde la nueva posición del cursor.

```
4610 GO SUB 90: GO SUB 80
4615 IF f=0 OR b>c1 THEN  GO SUB
 75: GO TO 4600
4620 LET c=b: GO SUB 95: INPUT "
repeat?";b$: IF b$<>"z" THEN  LE
T c=c+1: GO SUB 81: GO TO 4615
4625 GO TO 4600
```

La siguiente rutina combina los efectos de 'búsqueda', 'borrado' e 'inserción' para producir una 'búsqueda y sustitución'. Pero como esta denominación no cabe en el lugar de la pantalla destinado al menú especial, se la llama 'alteración'. La línea 4630 repite el trabajo realizado por 'búsqueda' pero preguntando por el nuevo texto que debe sustituir al antiguo. Tanto el viejo texto como el que le sustituirá se almacenan en el buffer de línea. Si al retornar, la bandera está colocada a cero, la búsqueda ha sido infructuosa y la rutina (75) así lo indicará antes de retornar al menú. Puede que no sea necesario en todos los casos sustituir un texto por otro, por eso la línea 4640 permite al usuario saltar sin abortar la búsqueda en caso necesario. La línea 4650 recurre al código máquina para borrar el texto viejo, después prepara a la misma rutina para que abra el espacio necesario para la inserción del nuevo texto y, por último, materializa la inserción. En el

Capítulo 6, páginas 78 y 79 hay un conjunto de diagramas que explican esta complicada operación.

```
463Ø GO SUB 9Ø: GO SUB 6Ø: PRINT
 AT 21,Ø;" NEW TEXT
          ";AT 21,12;: LET b=12:
 LET v2=c3: GO SUB 62: LET c4=c3
: LET c3=v2: IF c3=c4 THEN  GO T
O 4ØØØ
4635 GO SUB 81: IF f=Ø OR b>c1 T
HEN  GO SUB 75: GO TO 46ØØ
464Ø LET c=b: GO SUB 95: INPUT "
replace?";b$: IF b$<>"z" THEN  G
O TO 4665
465Ø LET m=c3-li: LET n=c4-c3: L
ET v1=c1-c-m: LET v2=te+c: LET v
3=v2+m: GO SUB 3Ø: LET v2=34999+
c1-m+n: LET v3=34999+c1-m: GO SU
B 35: LET v1=n: LET v2=te+c: LET
 v3=li+m: GO SUB 3Ø
```

Se adecuan los valores de las variables ‘c’ y ‘c1’ para el cambio, se llama a la rutina (95) para imprimir el texto editado en la pantalla y después la línea 4665 pregunta si se desean más sustituciones. La variable ‘c’ se desplaza un lugar para impedir que la búsqueda del mismo material continúe una y otra vez.

```
466Ø LET c1=c1-m+n: GO SUB 95
4665 INPUT "repeat?";b$: IF b$="
z" THEN  LET c=c+1: GO TO 4635
467Ø GO TO 46ØØ
```

Queda por explicar una de las seis opciones. La rutina ‘mueve’ permite que el material borrado se coloque en otro lugar del texto, pero para ello la etiqueta formada por el carácter CHR$ 255 (mencionada anteriormente) debe permanecer en su sitio, de lo contrario, la línea 48ØØ enviará el control al comienzo de la rutina de edición. La línea 481Ø permite que se inserte información casi al final del texto y las rutinas (3Ø) y (35) se utilizan para situar la información en su nuevo lugar, abriendo primero el espacio necesario y luego transfiriendo el material desde el buffer de línea.

```
48ØØ IF PEEK li<>255 THEN  GO TO
 4ØØØ
```

```
481Ø IF c=c1 THEN  LET c1=c1+1
482Ø LET b=PEEK 34ØØ1+256*PEEK 3
4ØØ2: LET v1=c1-c: LET v2=34999+
c1+b: LET v3=c1+34999: GO SUB 35
: LET v1=b: LET v2=c+te: LET v3=
34ØØ3: GO SUB 3Ø: LET c1=c1+b: G
O TO 4ØØØ
```

La opción para grabar el texto en cinta contiene instrucciones completas para su uso inclusive las de manipular los conectores. La línea 5Ø1Ø carga el valor de ‘c1’ en el buffer para que pueda recuperarse cuando se cargue de nuevo el texto en la computadora. La línea 5Ø2Ø graba el texto e incluye los dos bytes que contienen el valor de ‘c1’, después pregunta si se quiere verificar la grabación. La línea 5Ø3Ø controla la verificación y devuelve después el control al menú.

```
5ØØØ PRINT "Find a blank section
 of tape"'"Key a file ref. for t
he text": INPUT b$: PRINT b$''"K
ey C if correct": INPUT c$: CLS
: IF c$<>"c" THEN  GO TO 1
5Ø1Ø RANDOMIZE c1: POKE 34998,PE
EK q: POKE 34999,PEEK r
5Ø2Ø PRINT " PULL OUT THE EAR PL
UG ": SAVE b$CODE 34998,c1+2: IN
PUT "Key ""V"" to verify";z$: IF
 z$<>"v" THEN  GO TO 1
5Ø3Ø PRINT '''' FLASH 1;" REPLAC
E EAR PLUG ": PRINT ''''"Rewind
and play tape": VERIFY b$CODE :
GO TO 1
```

La computadora tiene que conocer cosas tales como cuántas líneas tiene que imprimir por página y si la impresora está conectada. La computadora es capaz de saber si la impresora está conectada sin necesidad de que se le diga, pero tiene que ser posible ‘mentir’ a la máquina para ahorrar papel. En la siguiente sección, por tanto, debe existir la opción de ‘desconectar’ la impresora para que el texto pueda escribirse en pantalla sin hacer una copia en papel hasta que no haya concluido la edición. La ‘desconexión’ se lleva a cabo fijando una bandera ‘p’ a cero y la ‘conexión’ fijándola a uno.

```
6ØØØ CLS : PRINT "Is the printer
 attached? (Y/N)": INPUT b$: IF
```

```
LEN b$<>1 OR (b$<>"y" AND b$<>"n
") THEN  GO TO 6ØØØ
```

Esta es la parte del programa que la computadora comienza a ejecutar cuando el programa se carga por primera vez. Las variables 'li' y 'te' se inicializan para estar preparadas para cuando se las necesite. La línea 6010 inicializa estas variables, desconecta la bandera 'p' de la impresora (a menos que se le indique lo contrario) y después fija el número de líneas por página 'x3' en 22 que es el valor por defecto. Termina dando la oportunidad usual de eliminar errores e introducir la información de nuevo. Antes de que aparezca el menú de opciones por primera vez, se inicializan algunas variables más: 'c2' es la longitud del buffer del texto pero 'q' y 'r' son más complejas. Cuando se utiliza RANDOMIZE, el número que sigue a la instrucción se divide por 256 y se calcula el resto. Dicho resto se almacena en la posición 23 670 y el cociente entero en el siguiente byte. Este es un procedimiento rápido, conveniente y que ahorra memoria para transformar un número en un formato de dos bytes tal y como se precisa a los efectos del lenguaje máquina. Las variables 'p' y 'q' contienen 23 670 y 23 671 respectivamente, y se utilizan para ejecutar el código máquina.

Si el usuario pide a la computadora que prepare el espacio para el texto, entonces sólo es preciso inicializar tres variables. Las variables 'c', 'c1' y 'b' toman el valor uno. No hay necesidad de borrar caracteres puesto que la computadora sólo interpreta los que se encuentran entre las posiciones 35 000 y 35 000 + c1. Al utilizar POKE 23 692, 9 se impide que la computadora se pare cada vez que completa la pantalla. La línea 7000 finaliza el programa.

```
6Ø1Ø LET te=35ØØØ: LET li=34ØØØ:
 LET x3=22: LET p=Ø: IF b$="y" T
HEN  LET p=1: PRINT '"How many l
ines per page?": INPUT x3: PRINT
 x3'"key C if correct": INPUT b$
: IF b$<>"c" THEN  CLS : GO TO 6
ØØØ
6Ø2Ø LET c2=15ØØØ: LET q=2367Ø:
LET r=23671: PRINT '"Key T to se
t up text space"'" CAUTION  dest
roys existing text": INPUT b$: I
F b$<>"t" THEN  GO TO 1
6Ø3Ø LET b=1: LET c=1: LET c1=1:
 POKE 23692,9: GO TO 1
7ØØØ PRINT AT 9,8;"END OF PROGRA
M"
```

Todo lo que resta del programa es el conjunto de subrutinas; la primera de las cuales es de uso muy frecuente. La cadena de la línea 2 tiene una longitud de 640 espacios:

2 LET a$="   ": RETURN

para introducir una línea semejante manténgase presionada la tecla espacio hasta que se consideren suficientes. Después ejecútese GO SUB 2 seguido de PRINT LEN a$. Si la máquina no respondiera '640', continúese ajustando la línea hasta que la respuesta sea exacta. En el programa SPECFILE se tendrá una línea idéntica, por tanto sería buena idea grabar ésta para ahorrar tiempo más tarde.

```
   1 GO TO 9ØØ
   2 LET a$="






                         ": RETURN
  1Ø IF INKEY$<>"" THEN  GO TO 1
Ø
  11 LET b$=INKEY$: IF b$="" THE
N  GO TO 11
  12 BEEP .ØØ3,Ø: RETURN
```

La rutina de formateo en la línea 3000 crea una línea a partir del buffer del texto de manera que no tenga más de 'y' + 1 caracteres. Dicha línea se justifica hasta la última palabra completa y así queda lista para ser impresa. Cuando la línea 20 tiene que situar el texto en la pantalla, la línea 21 comprueba si se trata de la línea vigésimo segunda. Lo que ocurre a continuación depende del valor que tenga la bandera de la impresora 'p'. Si 'p' contiene uno,

la computadora pide a la impresora que copie la pantalla, en caso contrario espera a que se pulse ENTER para comenzar otra página completa. La línea 23 suma uno a los contadores de líneas. La variable 'x1' cuenta las líneas en la pantalla y 'x4' cuenta el número de líneas impreso en papel desde que la impresora comenzó la última página.

```
  2Ø PRINT b$( TO y-x): LET x=y+
1
  21 IF x1=22 AND p=1 THEN  COPY
 : CLS : LET x1=Ø
  22 IF x1=22 AND p=Ø THEN  INPU
T c$: CLS : LET x1=Ø
  23 LET x1=x1+1: LET x4=x4+1: R
ETURN
```

Las dos rutinas que siguen no son tan complicadas como parece. En primer lugar, la línea 85 se asegura de que el contador de caracteres apunte hacia posiciones dentro del buffer del texto. Si se ejecuta RANDOMIZE con un número, la computadora lo trasforma en un formato de dos bytes y almacena el resultado en las posiciones 23 67Ø y 23 671. De este modo nos ahorramos el tener que dividir por 256, calcular el resto y después ejecutar POKE, dos veces, para almacenar el resto y el cociente. Puesto que 'p' y 'q' se han inicializado con 23 67Ø y 23 671, el efecto de la línea 3Ø es trasformar los valores de 'v1', 'v2' y 'v3' en un formato de 2 bytes y después cargar este valor en el lugar destinado al código máquina para prepararle para realizar un trabajo. El trabajo consiste en mover 'v1' bytes al lugar 'v2' desde la posición 'v3'.

```
  3Ø GO SUB 85: RANDOMIZE v1: PO
KE 6ØØØ1,PEEK q: POKE 6ØØØ2,PEEK
 r: RANDOMIZE v2: POKE 6ØØØ4,PEE
K q: POKE 6ØØØ5,PEEK r: RANDOMIZ
E v3: POKE 6ØØØ9,PEEK q: POKE 6Ø
Ø1Ø,PEEK r: RANDOMIZE USR 6ØØØØ:
 RETURN
  35 GO SUB 85: RANDOMIZE v1: PO
KE 6ØØ15,PEEK q: POKE 6ØØ16,PEEK
 r: RANDOMIZE v2: POKE 6ØØ18,PEE
K q: POKE 6ØØ19,PEEK r: RANDOMIZ
E v3: POKE 6ØØ23,PEEK q: POKE 6Ø
Ø24,PEEK r: RANDOMIZE USR 6ØØ14:
 RETURN
```

La siguiente rutina mueve el cursor en la pantalla. Para llamar a la rutina del teclado se recurre a la línea 11 en lugar de a la 1Ø (que es donde comienza) de forma que la computadora no espere a que se haya levantado el dedo del último carácter pulsado para cargar un carácter en b$. Esta sutileza proporciona además la posibilidad de repetición de una tecla si se mantiene presionada. La línea 4Ø produce el cursor. Las coordenadas en la pantalla 'm' y 'n' se fijan para que el valor corresponda a la parte izquierda en la mitad de la pantalla, la función SCREEN$ posibilita a la computadora re-escribir el carácter con el FLASH accionado. El mismo carácter se imprime en el mismo lugar por la línea 41, pero en esta ocasión sin intermitencia. No obstante, antes de todo, la línea 41 utiliza a la rutina del teclado para comprobar si ha habido pulsaciones.

```
4Ø LET c$=SCREEN$ (m,n): PRINT
FLASH 1;AT m,n;c$
41 GO SUB 11: PRINT AT m,n;c$
```

Las cuatro líneas que siguen ajustan 'm' y 'n' para hacer que el cursor se mueva en la dirección de las teclas con flechas (desde el 5 hasta el 8).

```
  42 IF b$="5" THEN  LET n=n-1:
LET c=c-1
  43 IF b$="6" THEN  LET m=m+1:
LET c=c+32
  44 IF b$="7" THEN  LET m=m-1:
LET c=c-32
  45 IF b$="8" THEN  LET n=n+1:
LET c=c+1
```

Las líneas restantes comprueban que la nueva posición del cursor está dentro de los límites de la pantalla antes de devolver el control al programa principal.

```
  46 IF n<Ø THEN  LET n=n+32: LE
T m=m-1
  47 IF n>31 THEN  LET n=n-32: L
ET m=m+1
  48 IF m<Ø THEN  LET c=c+32: LE
T m=m+1: GO TO 48
  49 IF m>19 THEN  LET m=m-1: LE
T c=c-32: GO TO 49
  5Ø RETURN
```

La siguiente rutina es una versión reducida de la rutina de entrada de texto y se utiliza para las inserciones, búsquedas y sustituciones. Las primeras líneas son muy directas. El CHR$ 13 es el carácter ENTER.

```
  6Ø PRINT AT 2Ø,Ø;" KEY TEXT "
  61 PRINT AT 21,Ø;: LET c3=li:
LET b=Ø
  62 GO SUB 1Ø: LET a=CODE b$: I
F a=13 THEN  RETURN
```

Puesto que la computadora trata el texto como una cadena el uso de comillas dentro del mismo podría producir problemas, por este motivo se ha definido el CHR$ 144 para sustituir a las comillas. La línea 63 realiza el cambio de unas comillas por otras y envía los caracteres DELETE a la línea 7Ø para que allí se los procese. La línea 64 rechaza los caracteres no permitidos. La existencia de las comillas definidas por CHR$ 144 obliga a complicar esta línea. La variable 'c3' contiene el siguiente lugar disponible en el buffer del texto, la línea 66 almacena en este lugar (mediante POKE) el carácter siguiente y lo imprime en la pantalla antes de preparar las variables para el carácter que sigue. Es prácticamente impensable que alguien desee insertar más de 1ØØØ caracteres de una vez, pero por si acaso, la línea 67 se ocupa de impedir que el buffer de línea pueda invadir el espacio del buffer del texto a partir de la posición 35 ØØØ.

```
  63 IF CODE b$=34  THEN  LET a=
144: LET b$=CHR$ a:
  64 IF a=12 THEN  GO TO 7Ø
  65 IF a<32 OR (a>127 AND a<>14
4) THEN  GO TO 62
  66 PRINT b$;: POKE c3,a: LET b
=b+1: LET c3=c3+1: IF b=32 THEN
 LET b=Ø: POKE 23692,9
  67 IF c3=34997 THEN  RETURN
  68 GO TO 62
```

La posibilidad de corregir el texto suplementario se reduce a borrar caracteres y volverlos a escribir de nuevo, pero se considera suficiente para lo que se pretende. La única dificultad que podría surgir, es la posibilidad de borrar más allá del principio del buffer y de que esto no pueda suceder se encarga la línea 7Ø.

```
 7Ø LET c3=c3-1: IF c3<li THEN
LET c3=li: GO TO 62
 71 LET b=b-1: IF b<Ø THEN  LET
b=31
 72 PRINT AT 21,b;"_ ";AT 21,b;
: GO TO 62
```

Después de búsquedas infructuosas se llama a la rutina (75) que se ocupa de escribir en la pantalla un mensaje adecuado.

```
 75 PRINT AT 2Ø,Ø;" FILE END
                  ": RETURN
```

La instrucción RANDOMIZE se utiliza varias veces en las próximas líneas. De nuevo, ahorra espacio convirtiendo a datos numéricos en un formato adecuado para ocupar dos posiciones de memoria. Esta rutina prepara la llamada a la rutina FIND escrita en código máquina, la llama y de acuerdo con el resultado de la búsqueda otorga valores a las variables 'f' y 'b'.

```
  8Ø GO SUB 6Ø
  81 RANDOMIZE li: POKE 6Ø137,PE
EK q: POKE 6Ø138,PEEK r: RANDOMI
ZE te+c2: POKE 6Ø139,PEEK q: POK
E 6Ø14Ø,PEEK r: RANDOMIZE c3: PO
KE 6Ø141,PEEK q: POKE 6Ø142,PEEK
 r: RANDOMIZE USR 6ØØ28
  82 LET f=PEEK 6Ø143: LET b=PEE
K 6Ø135+256*PEEK 6Ø136-te: RETUR
N
```

La rutina que sigue no tiene ningún interés especial, se ocupa tan solo de mantener a 'c' dentro de los límites del buffer del texto.

```
 85 IF c<1 THEN  LET c=1
 86 IF c>c1 THEN  LET c=c1
 87 RETURN
```

La rutina (9Ø) posibilita la ejecución del código máquina mirando desde el comienzo del texto o desde la posición del cursor. En las posiciones 6Ø 135 y 6Ø 136 es donde el programa en código máquina encuentra la información que le indica en qué parte del texto debe comenzar.

```
  9Ø INPUT "from cursor?";b$: RA
NDOMIZE te: IF b$="z" THEN  RAND
OMIZE te+c
  91 POKE 6Ø135,PEEK q: POKE 6Ø1
36,PEEK r: RETURN
```

En lo que resta, es donde realmente se realiza el trabajo. La rutina (95) carga 'v1' bytes desde la posición 'v3' en el buffer del texto, a partir del lugar 'v2' que es siempre alguna posición dentro de la cadena que se define en la línea 2. La línea 96 llama a la rutina (3Ø) para que realice el movimiento, limpia la pantalla y después va a la línea 2 para enterarse de lo que ésta ha recibido. Cuando vuelve, imprime a$ en la pantalla y añade el cursor intermitente.

```
  95 LET v1=64Ø: LET v2=23779: L
ET v3=c+3468Ø: IF c<321 OR c>c1-
32Ø THEN  GO SUB 97
  96 GO SUB 3Ø: CLS : GO SUB 2:
PRINT a$: LET m=1Ø: LET n=Ø: PRI
NT  FLASH 1;AT m,n;SCREEN$ (m,n)
: RETURN
```

Si la línea 95 detectase que el bloque de texto estaba muy cerca del principio o del final, el bloque seleccionado no rellenaría los 640 espacios que componen la pantalla. La línea 97 llama a la última rutina, la que rellena la línea 2 con 640 espacios. Las líneas 98 y 99 calculan cuantos de esos espacios deben rellenarse con caracteres.

```
  97 GO SUB 1ØØ
  98 IF c<321 THEN  LET v1=c+319
: LET v2=241ØØ-c: LET v3=te: RET
URN
  99 LET v1=32Ø+c1-c: LET v2+237
79: LET v3=c+3468Ø: RETURN
```

Aquí se tiene la rutina que acabamos de mencionar. La primera dirección en el interior de la cadena de la línea 2 es 23 779 y la última 24 418. La rutina en código máquina que empieza en la posición 6Ø 151 rellena estas posiciones con el carácter espacio, cuyo código es 32.

```
 1ØØ RANDOMIZE 23779: POKE 6Ø152
,PEEK q: POKE 6Ø153,PEEK r: RAND
```

```
OMIZE 24418: POKE 6Ø155,PEEK q:
POKE 6Ø156,PEEK r: POKE 6Ø158,32
: RANDOMIZE USR 6Ø151: RETURN
```

La parte del programa escrita en BASIC ha concluido, *pero no debe ejecutarse sin que el código máquina ocupe su lugar,* de lo contrario el microprocesador entrará en la memoria y nunca volverá. Pulsar BREAK no solucionaría el problema. La única solución sería apagar la computadora y conectarla de nuevo, pero se perdería el programa. La parte del programa escrita en lenguaje máquina se ofrece en dos formatos diferentes en el Capítulo 5. Para los lectores que disponen de ensamblador tienen los códigos nemónicos y para quiénes no tengan ensamblador se dispone de una lista numérica.

5 INSTRUCCIONES OPERATIVAS DE SPECTEXT

En la figura 5.1 se da un diagrama operativo de SPECTEXT. Cuando el programa se carga por primera vez en la computadora precisa alguna información y por ello realiza algunas preguntas. Por el momento, pulse la tecla 'N' y después ENTER cuando se le pregunte sobre la impresora; después pulse 'T' para reservar espacio para trabajar. Pulse la tecla ENTER de nuevo para confirmar su solicitud y la computadora se dirigirá hacia el 'menú de opciones'. Para seleccionar una opción pulse la tecla correspondiente (véase Fig. 5.2), no hay necesidad de pulsar ENTER en esta ocasión. El resto del capítulo se dedicará a estas opciones. Algunas disponen a su vez de varias subsecciones, cada una de las cuales se tratará por separado mediante un párrafo, de manera que pueda utilizarse el capítulo como referencia. Esta forma de explicación es especialmente adecuada para los principiantes con objeto de familiarizarlos con el programa, sobre todo si no lo han utilizado durante un cierto tiempo.

ENTRADA DEL TEXTO

Al presionar la tecla '1' mientras se está visualizando el menú, la pantalla se borra a excepción de una pequeña marca en la esquina inferior izquierda. En ese lugar es donde empezará a aparecer el texto cuando se teclee. La escritura del texto no requiere comentarios. La tecla SHIFT permite el acceso a las mayúsculas y la tecla SYMBOL SHIFT posibilita el acceso a algunos caracteres que aparecen en rojo sobre las teclas. Las teclas que ocasionasen problemas sobre la pantalla se rechazan por el propio programa, pero es muy improbable que algún usuario las eche en falta. Puede surgir un problema al tener pulsada la tecla SHIFT. Si se pulsa la tecla BREAK el programa interrumpirá su ejecución. Al final del capítulo se da un remedio sencillo contra esta eventualidad. Entre tanto, tenga cuidado cuando introduzca varias palabras seguidas en mayúsculas. Si se produce una equivocación funcionará el procedimiento normal de pulsar SHIFT/Ø, por tanto la computadora se comporta como una máquina de escribir con autocorrección. No hay necesidad de intentar que cada

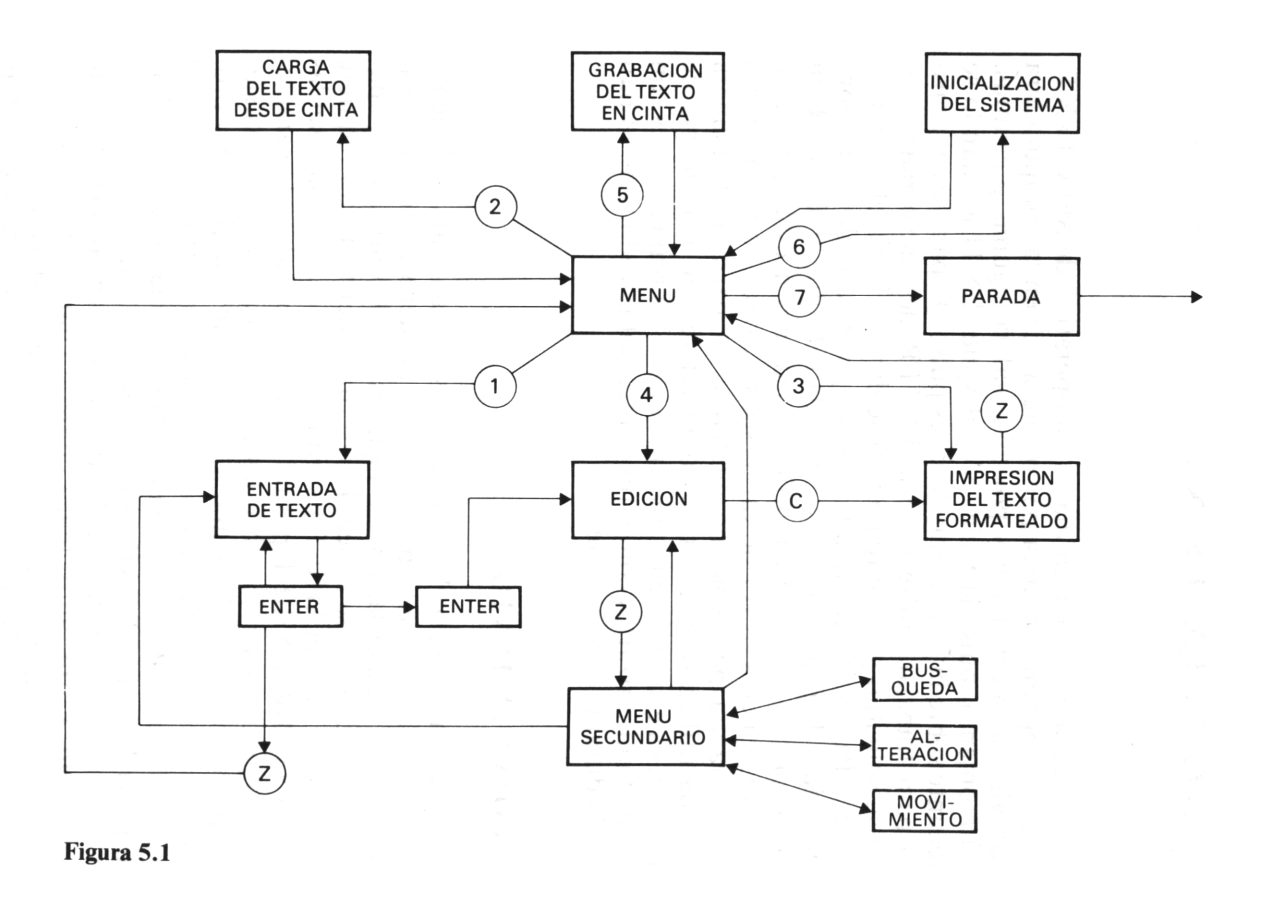

CARGA DEL TEXTO DESDE CINTA
GRABACION DEL TEXTO EN CINTA
INICIALIZACION DEL SISTEMA
2
5
6
MENU
7
PARADA
1
4
3
Z
ENTRADA DE TEXTO
EDICION
C
IMPRESION DEL TEXTO FORMATEADO
ENTER
ENTER
Z
BUS-QUEDA
MENU SECUNDARIO
AL-TERACION
Z
MOVI-MIENTO

Figura 5.1

Opción 1 — escritura de texto

ENTER	posibilita la entrada de marcadores para un correcto formateo; véanse las indicaciones en pantalla ENTER mueve el control a la opción 4 "Z" da el menú
SHIFT/Ø	borra el último carácter

Opción 2 — carga del texto desde cinta

siga las instrucciones que aparecen en pantalla

Opción 3 — impresión del texto

"Z"	se abandona el menú

Opción 4 — edición del texto

"P"	se avanza una página
"Q"	se retrocede una página
5 a 8	control del cursor
"E"	edición en la posición del cursor "A" insertar después del cursor (siga las instrucciones que aparecen en pantalla) "D" borrado — sitúe el cursor sobre el último carácter que debe borrarse y pulse de nuevo "D" para borrar
"C"	imprime el texto desde la posición del cursor
"Z"	menú adicional de seis opciones. Pulsando ENTER se pasa a la siguiente opción y pulsando "Z" se selecciona la opción que aparece en pantalla

1 escritura	se puede continuar escribiendo desde el cursor
2 edición	retorna a la rutina de edición
3 búsqueda	encuentra una frase específica, con ENTER se repite
4 alteración	sustituye una frase específica por otra alternativa, con "Z" se puede introducir el nuevo texto y con "Z" de nuevo se puede repetir
5 mueve	inserta el texto recién borrado, antes de la posición del cursor
6 menú	vuelve al menú principal

Opción 5 — grabación del texto en cinta

siga las instrucciones que aparecen en pantalla

Opción 6 — mantenimiento

siga las instrucciones que aparecen en pantalla

Figura 5.2

línea comience con una nueva palabra; la máquina se encargará de este ajuste cuando llegue el momento de la impresión. Se ahorran espacio y pulsaciones mediante un sistema de marcadores que se utilizan para indicar a la computadora donde comenzar una nueva línea, un nuevo párrafo o una nueva página. Para introducir uno de estos marcadores, pulse la tecla ENTER y dispondrá de cinco nuevas opciones. La tecla '1' colocará un marcador de línea ▀ en la pantalla, mediante '2' se selecciona el marcador de párrafo ■, con '3' se le indica a la computadora que rellene el resto de la página con líneas en blanco y empiece otra nueva. Quedan dos opciones más. La tecla 'Z' permite detener la escritura durante unos momentos para volver al menú y pulsando ENTER nos situaremos en la opción de edición para corregir lo que se haya escrito. Se tiene que pulsar ENTER después de la tecla de cada opción excepto en la última. Si se utilizan las opciones 'Z' o ENTER la computadora recordará en qué parte del texto nos encontrábamos y volverá al mismo sitio, a menos que se le indique el deseo de empezar en otra parte.

La pantalla se rellena en un instante y aunque podría parecer alarmante ver como desaparece el texto cuidadosamente escrito por la parte superior, no hay necesidad de preocuparse, puesto que todos los caracteres están almacenados en la memoria. La memoria se acabará completando. Cuando esto suceda se escuchará un tono de aviso y la computadora no aceptará la entrada de nuevo texto hasta que no se haya registrado en cinta el contenido del buffer del texto y se haya vuelto a ejecutar el programa. Si el programa se ha inicializado como se indica en el Capítulo 4, se podrán procesar 15 000 caracteres. Esto es equivalente a alrededor de 3000 palabras inglesas, lo que puede considerarse un volumen de texto aceptable. Sería prudente dejar la posibilidad de entrar en edición, hasta que no se alcance la opción que la permite, de este modo la edición será completa.

CARGA DEL TEXTO DESDE CINTA

Para utilizar esta opción será preciso conocer el nombre del fichero que desea cargarse. Hay una oportunidad para confirmar el nombre que se ha escrito, pero si no se decide a escribirlo de nuevo, la máquina retornará al menú principal. De este modo, esto puede resultar útil para salir de la rutina si se ha seleccionado por error. Una vez que se ha confirmado el nombre, la computadora proporcionará todas las instrucciones necesarias a través de la pantalla e intentará cargar el bloque de texto desde la cinta. Si hubiera dificultades (quizás provocadas por un fallo en la cinta) el

programa las detectará. Para salir presione SHIFT y BREAK conjuntamente y después GO TO 1 seguido de ENTER, con lo cual se volverá al menú. Si en lugar de GO TO 1 se pulsa RUN se perderá toda la información almacenada. Si le sucediera esto, pulse SHIFT y BREAK conjuntamente, seguido de GO TO 3. Los usuarios más experimentados serán capaces de recuperar la información dando un adecuado valor a 'c1', el correspondiente al último carácter introducido en el buffer del texto antes de ejecutar GO TO 3. Aquí es donde la computadora dará por concluida la operación si el texto se ha cargado con éxito. En este momento se puede añadir nuevo texto, editarlo, volverlo a grabar o imprimirlo.

IMPRESION DEL TEXTO

Esta es otra opción a la que se puede acceder desde el menú o desde la cuarta opción (edición). Si se pulsa la tecla '3' desde el menú, se escribirá el texto desde el principio en la pantalla. Si se accede a esta opción desde la edición, el texto se escribirá desde la posición del cursor. La computadora tomará el texto correspondiente a una línea y a continuación lo escribirá en un formato muy acabado con un ligero margen izquierdo y sin fraccionar palabras al final de la línea. Los marcadores se interpretarán pero no se escribirán en la pantalla. Si se le comunica a la computadora que la impresora está conectada, realizará una copia en papel del texto ya formateado sin necesidad de más intervención. Si la máquina piensa que la impresora no está conectada permanecerá en espera al finalizar cada página y no pasará a la siguiente hasta que no se pulse ENTER. Durante la escritura de una página se puede abandonar la rutina pulsando la tecla 'Z'.

EDICION DEL TEXTO

Se puede seleccionar esta opción pulsando '4' cuando se está ante el menú o bien, abandonando la opción 1 (entrada de texto). Las teclas 'Z' y ENTER nos permiten movernos rápidamente dentro de estas dos opciones. Cuando se está introduciendo texto se puede acceder a la opción 4 presionando para ello dos veces la tecla ENTER. Mientras se está editando se puede continuar escribiendo desde la posición del cursor pulsando para ello dos veces la tecla 'Z' seguida de ENTER. Esta opción es la más extensa de todas y dispone de varias subsecciones. Con lo primero que debemos acostumbrarnos es con el control del cursor.

Control del cursor

Cuando se selecciona la opción 4 aparece el texto con el cursor a la izquierda y en la mitad de la pantalla. Esta posición puede considerarse como la casa del cursor. Puede moverse a través de toda la pantalla, pero al concluir la edición siempre retorna aquí. Las teclas con flechas mueven el cursor en la dirección de las mismas, pero la computadora impide que el cursor pueda salirse de la página. Las teclas 'P' y 'Q' sirven para pasar de página. 'P' avanza y 'Q' retrocede una página. Siempre se solapa una línea al pasar de página para que la lectura del texto pueda efectuarse con continuidad. Cuando la máquina alcanza el final del texto, éste se escribe en la pantalla de tal modo, que el principio o el final del texto se corresponda con la posición de casa del cursor.

Otras funciones durante la edición

Cuando se pulsa la tecla 'Z' desde la opción de edición, se abre una extensa variedad de nuevas funciones a nuestra disposición. Las funciones aparecen en un menú con un formato especial que presenta una opción cada vez sobre una pequeña caja situada en el ángulo inferior izquierdo de la pantalla. Pulsando la tecla ENTER se rechaza la opción en curso y aparece la siguiente en la misma caja. Para seleccionar una opción debe pulsarse la tecla 'Z' y después ENTER, cuando la misma aparece en la caja. Hay seis opciones que se escriben alternativamente en blanco y negro para llamar la atención.

Escritura

Mediante esta opción se puede continuar escribiendo desde la posición del cursor. El texto se imprime de modo que se alinie con el primer carácter que va a escribirse, el cual aparecerá en la parte izquierda de la línea inferior.

Edición

Devuelve al usuario a la rutina de edición.

Búsqueda

Mediante esta rutina se puede localizar material en el interior del texto. La computadora debe saber si la búsqueda debe efectuarse desde la posición del cursor o por el contrario desde el principio del texto, así como, qué se desea localizar. Una vez que se ha informado a la computadora de lo que necesita, comienza la búsqueda del material y, o bien da un mensaje de fin de fichero, o bien escribe texto en la pantalla. El mensaje de fin de fichero sig-

nifica que se ha concluido el texto sin haber localizado el material. Si se imprime texto corresponderá a la primera ocurrencia del material en el buffer del texto, pero podría haber más. La computadora nos da la oportunidad de continuar con la búsqueda hasta el final del texto. En este caso cada ocurrencia del material se escribirá cuando se encuentre desde la posición de casa del cursor. Con ENTER se continúa la búsqueda y con 'Z' se retorna al menú.

Alteración

Esta opción es parecida a la anterior, pero el usuario puede sustituir el material buscado por otro bloque de texto. Sobre la pantalla se escriben instrucciones completas para el uso de esta opción. Durante la ejecución el usuario puede decidir en que ocurrencias efectuar la sustitución y cuales otras dejar inalteradas.

Mueve

El funcionamiento de esta rutina depende de la situación de una bandera. Funcionará tan solo si la bandera está puesta. Cuando se borra una porción del texto se transfiere a una zona especial de la memoria. Con esta opción se puede reubicar este material en cualquier lugar del texto tantas veces como se desee. El texto se inserta en la posición del cursor. Las inserciones adicionales se llevan a cabo colocando el cursor en el lugar deseado y seleccionando de nuevo esta opción. Sin embargo, tan pronto como se utiliza otra función existe el peligro de perder el material almacenado y que la opción no vuelva a funcionar.

Menú

Devuelve el control al menú principal.

Inserción y borrado

Estas funciones pueden utilizarse dondequiera que esté el cursor. Se seleccionan pulsando la tecla 'E'. Todavía se dispone de otro tipo de menú como respuesta a esta selección. Tiene dos opciones, 'A' para insertar material y 'D' para borrar. Es preciso confirmar la opción pulsando ENTER. Las alteraciones sucederán sobre la posición del cursor y éste se mueve al final de las mismas del cambio. Después el texto se escribe de modo que el cursor aparezca en su casa. Con 'A' se inserta nuevo texto inmediatamente después del cursor. El texto se introduce en la parte inferior de la pantalla y puede corregirse mediante la tecla DELETE. Cuando se pulsa ENTER el nuevo material se inserta y el

texto se vuelve a escribir en su nueva versión. Cuando se pulsa la tecla 'D', se subraya el carácter correspondiente a la posición del cursor. Con las teclas con flechas se puede mover ahora el cursor hasta otro lugar posterior en el texto. Tan pronto como se pulse de nuevo 'D', los caracteres comprendidos entre el carácter subrayado y la posición actual del cursor, ambos inclusive, serán borrados y el texto se volverá a escribir en la pantalla de forma que la posición del cursor corresponda a su casa, es decir en la parte izquierda y la mitad de la pantalla.

Impresión

Hay otra posibilidad antes de abandonar la opción 4 del menú. La tecla 'C' o LPRINT obligará a la computadora a producir una copia del texto en la pantalla y la impresora desde la posición del cursor hasta el final del texto, o bien hasta que se pulse la tecla 'Z'. Sólo se realiza una copia en papel si previamente se le ha dicho a la computadora que la impresora está conectada. Para conectarla debe recurrirse a la opción 6 del menú.

GRABACION EN CINTA DEL TEXTO

Todo lo que se necesita en esta opción es un nombre para el texto que va a grabarse. La computadora le dará todas las instrucciones necesarias y cuando se complete la grabación volverá al menú. Se puede verificar la grabación comparando el registro en cinta con el texto almacenado en la memoria. Después de la grabación el texto permanecerá en la memoria y será preciso recurrir a la opción siguiente para preparar la máquina para introducir nuevo texto.

PREPARACION DEL SISTEMA

Se necesita recurrir a esta opción una vez por lo menos cada vez que se utiliza SPECTEXT. De hecho se visita esta opción automáticamente cada vez que se carga el programa, pero puede volver a visitarse tantas veces como sea necesario. Puede utilizarse la opción como una especie de botón que permite conectar y desconectar la impresora. Si se desea contemplar el texto tal y como será impreso sin usar para ello la impresora, se puede desconectar desde esta opción.

Si se le dice a la computadora que la impresora está conectada, querrá saber algunas cosas más relativas al formato. Conteste todas las preguntas y la máquina le llevará de nuevo al menú.

CONSEJOS OPERATIVOS

Hay unas pocas maneras de interrumpir bruscamente el sistema, por ejemplo pulsando simultáneamente las teclas SHIFT y BREAK. La cura es sencilla, pulse SHIFT y '1' conjuntamente y después ENTER. De este modo se suprimen las pulsaciones dadas antes de que se produjera el error. Con GO TO 1 y después ENTER se volverá al menú y la opción 4 nos situará sobre el último carácter. La tecla 'Z' nos permitirá escribir de nuevo.

Como ocurre con todos los programas de este libro hay solo unas pocas teclas para controlar las funciones. La tecla 'Z' realiza la mayor parte de las funciones de control que no son realizadas por ENTER. La única tecla que ejecuta una función de control y que no se facilita su uso con mensajes en pantalla es la tecla 'E', sirve para la inserción y borrado de caracteres en la rutina de edición. El sistema es muy sencillo de manejar. La mayor parte de la gente que ha utilizado SPECTEXT ha descubierto sus propias formas de satisfacer sus particulares necesidades. No es posible desarrollarlas aquí, pero las siguientes sugerencias ayudarán considerablemente.

Si se ha introducido una letra incorrecta y no nos hemos dado cuenta en principio, será necesaria su sustitución. Esto puede realizarse en un solo paso colocando el cursor sobre el error y seleccionando después la opción de escritura desde el menú de edición. Ahora puede escribirse el nuevo carácter y dos pulsaciones ENTER nos devolverán a la rutina de edición para continuar las correcciones.

Cuando se necesite ir bastante hacia el final del texto, la manera obvia es utilizando las teclas con flechas. Sin embargo, utilizando la tecla 'P' se ganará tiempo avanzando de página en página. De forma análoga no hay necesidad de pulsar repetidamente '8' para ir desde el extremo izquierdo de una línea al extremo derecho, mediante '5' se conseguirá el propósito en menos pulsaciones. La interrogación al final del texto muestra la siguiente posición y desaparecerá en cuanto se seleccione la opción de escritura.

Cuando se esté utilizando la opción de búsqueda (SEEK), la rutina ignorará cualquier palabra al principio de una frase si se la ha pedido que localice una palabra que está escrita íntegramente con minúsculas. Un modo de encontrar todas las ocurrencias consiste en localizar todo excepto la primera letra. Preguntando por 'assette' se localiza tanto 'Cassette' como 'cassette' y sus plurales respectivos. Si la palabra buscada tuviera un plural extraño convendría utilizar, tan solo las letras comunes a ambos.

La opción mueve (MOVE) puede utilizarse para diseminar

una frase estándar o una cláusula a lo largo de un documento. La frase necesita teclearse sólo la primera vez que aparezca y después debe borrarse. Seleccionando la opción de mover, se podrá situar a esta frase no solo en el lugar primitivo, sino también en cualquier otro lugar del texto que se desee, tan solo tardará la inserción una fracción de segundo. El otro uso obvio de esta opción es para reestructurar un párrafo alterando el orden de las frases que lo componen.

Es peligroso pero interesante suprimir todas las comprobaciones que realiza la opción de búsqueda y sustitución (ALTER). Si por ejemplo se considera necesario sustituir en todo un texto la palabra 'entre' por las palabras 'en medio de', podría cambiarse en el texto la palabra 'entretener' por 'en medio detener' que obviamente no es lo que se pretendía. No obstante es interesante ver el programa modificado en acción y para ello solo se necesitan pequeños cambios. En la línea 4640 necesitamos añadir 'GO TO 4650' después de 'GO SUB 45' y antes de 'INPUT..., etc.'. Para terminar el cambio se necesita una nueva línea:

```
4664 LET c=c+1: GO TO 4635
```

Otra modificación para extender el buffer del almacenamiento del texto podría considerarse útil y es incluso más simple. La longitud del buffer se fija en 15 000 caracteres por la línea 6020. Este volumen puede ser almacenado en cassette en poco más de un minuto, lo que se considera un tiempo razonable. La dimensión máxima del buffer es 24 000 caracteres, lo que significaría 2 minutos de carga. El tiempo adicional es inaceptable. Pero hay otra consideración importante. Un fallo en el suministro de energía (tal vez simplemente porque Maribel desenchufa accidentalmente al mover una silla) significa la pérdida de 24 000 caracteres que habría que volver a teclear. Si quiere correr el riesgo cambie la línea 6020 de modo que quede 'LET c2 = 24000'. No se pretenda incrementar esta cifra o de lo contrario se podría machacar el código máquina.

El hecho de que estos programas puedan modificarse fácilmente debe incitar al lector a la experimentación para adaptarlos a sus propias necesidades. Algunas partes del programa se interrelacionan con otras pero, siempre que se conserve la versión original no hay problema si falla el experimento. El estilo de presentación se ha desarrollado para permitir este tipo de experimentos y hacer que el lector se sienta al mando de la computadora.

SPECTEXT subrutinas y variables

El programa se ha construido sobre un pequeño número de variables, muchas de las cuales se han utilizado para más de una

tarea. La lista que sigue contiene las variables importantes. Las que no se incluyen realizan pequeñas tareas y tiene importancia tan solo en las pocas líneas en que aparecen.

Variables

a$	—pantalla del texto
b	—contador de columna para la pantalla
c	—carácter en curso
c1	—lugar hasta el cual se ha rellenado el buffer
c2	—límite del buffer del texto (15 000 caracteres)
c3	—fin del buffer de línea
d	—número de línea en la pantalla
f	—bandera de propósito general
g	—contador utilizado en la edición
j	—contador de bucle
m y n	—coordenadas en la pantalla
p	—bandera de la impresora
q y r	—se utilizan junto con RANDOMIZE para generar números en formato de dos bytes
x a x4	—variables para el formateo
te	—principio del buffer (35 000)
li	—principio del buffer de línea (34 000)

Subrutinas

Rutina 2—fija la variable a$ con el contenido de la cadena de la línea 2. El texto se carga desde el buffer por medio de una rutina en código máquina.

Rutina 10—lee el teclado y carga la primera pulsación en b$.

Rutina 20—escribe el texto formateado en la pantalla y la impresora, teniendo en cuenta el número de líneas dentro de la página.

Rutina 30—esta es una rutina disparadora del código máquina que mueve un bloque de información de v1 bytes hacia abajo desde la dirección v3 a la dirección v2.

Rutina 35—similar a la anterior, pero realizando el movimiento en sentido contrario, es decir, hacia arriba.

Rutina 40—control del cursor para la edición.

Rutina 60—entrada de información suplementaria para búsquedas.

Rutina 75—imprime el mensaje de fin de fichero.

Rutina 80—inicializa la rutina en código máquina SEARCH.

Rutina 90—fija el primer carácter para la rutina SEARCH (búsqueda), posición del cursor o principio de texto.

Rutina 95—prepara el código máquina para mover una sección del texto desde el buffer del texto hasta la cadena de la línea 2.

Rutina 100—rellena la cadena de la línea 2 con espacios y limpia parte de la pantalla.

6 CODIGO MAQUINA DE SPECTEXT

Cuando el BASIC resulta demasiado lento hay que recurrir al lenguaje máquina. La zambullida tiene que ser rápida, pero una vez realizada se tiene la impresión de que el tema ya era conocido. Esto se debe probablemente a que nos encontramos con los mismos problemas que cuando empezamos con el BASIC. El BASIC parece también muy complicado al principio y lo mismo sucede con este nuevo lenguaje. No obstante, el problema es menos agudo en esta ocasión por varios motivos. En primer lugar el BASIC es distinto que el lenguaje máquina pero no tan diferente, el lenguaje máquina es más simple. El número de instrucciones es elevado pero muchas de ellas realizan esencialmente el mismo trabajo pero con diferentes registros internos del microprocesador. Solo utilizaremos unas pocas instrucciones de modo que todo resulte más sencillo. Sin embargo, la mayor contribución no está en reducir el número de instrucciones con el que vamos a trabajar, sino en la forma en que se va a tratar el tema.

En el Capítulo 4 se adelantó que las mayores dificultades estaban en la localización y corrección de los errores que siempre son inevitables al escribir programas. No se dispone de mensajes de error a menos que se introduzcan dentro del programa y cuando un programa no funciona, frecuentemente no puede pararse. La solución está en escribir el código máquina en BASIC y realizar todas las comprobaciones y modificaciones en un lenguaje de alto nivel. Tan solo cuando se han realizado todas las modificaciones y el programa funciona se traduce al lenguaje máquina. Para que este sistema funcione hay que poner de manifiesto tanto las identidades como las diferencias entre ambos lenguajes. En la tabla 6.1 se dispone de una lista de las instrucciones máquina que serán necesarias para escribir las rutinas. En la segunda columna están las instrucciones BASIC que realizan la misma función. Si los programas se escriben en BASIC con estas instrucciones, las correcciones y la traducción pueden realizarse con mucha facilidad.

Sería posible escribir totalmente el procesador de textos en lenguaje máquina. Las ventajas de utilizar esta técnica no eliminan la existencia de algunas desventajas. Si se utiliza el BASIC para la mayor parte del programa, se podrá modificar con mucha

Tabla 6.1. Equivalentes en BASIC de las instrucciones máquina

add 5	LET A+A+5
add l	LET A=A+L
sub 5	LET A=A-5
sub l	LET A=A-L
adc hl, bc	LET HL=HL+BC
sbc hl, bc	LET HL=HL-BC
machine code stack	DIM Z(100): LET SP=1
push hl	LET Z(SP)=HL: LET SP=SP+1
pop hl	LET SP=SP-1: LET HL=Z(SP)
ldir	FOR J=0 TO BC-1: POKE DE+J, PEEK HL+J: NEXT J: LET HL=HL+BC: LET DE=DE+BC: LET BC=0
lddr	FOR J=BC-1 TO 0 STEP-1: POKE DE+J, PEEK HL+J: NEXT J: LET HL =HL-BC: LET DE=DE-BC: LET BC=0
ld a,5	LET A=5
ld a, (35000)	LETA=PEEK 35000
ld, a (bc)	LET A=PEEK BC
ld (35000),a	POKE 35000, A
ld hl, 35000	LET HL=35000
ld hl, (35000)	LET HL=PEEK 35000+256*PEEK 35001
ld hl, bc	LET HL=PEEK BC+256*PEEK (BC+1)
ld (35000), hl	POKE 35001, INT (HL/256): POKE 35000, HL-(PEEK 35001*256)
inc a	LET A=A+1
dec a	LET A=A-1
inc hl	LET HL=HL+1
dec hl	LET HL=HL - 1
jp 35000	GOTO a line number
jp z, 35000	IF A=0 THEN GO TO a line number
jp p, 35000	IF A >=0 THEN GO TO a line number
jp m, 35000	IF A < 0 THEN GO TO a line number
jp nz, 35000	IF A < > 0 THEN GO TO a line number
call 35000	GO SUB a line number
call z, 35000	IF A=0 THEN GO SUB a line number
call nz, 35000	IF A < > 0 THEN GO SUB a line number
ret	RETURN
ret z	IF A=0 THEN RETURN
ret nz	IF A < > 0 THEN RETURN

mayor facilidad. Disponer de los dos lenguajes ofrece las ventajas de ambos, reduciendo las desventajas a su mínima expresión. Otra gran diferencia entre ambos lenguajes es el número de variables que pueden utilizarse. Los únicos nombres que pueden utilizarse al trabajar con los códigos de operación del Z80 OP son los correspondientes a los registros del microprocesador y de éstos, tan solo siete pueden utilizarse con facilidad. Los nombres de los registros son: a, b, c, d, e, h y l. Estos registros pueden utilizarse por separado para almacenar números comprendidos entre Ø y 255. Para los números comprendidos entre Ø y 65 535 pueden utilizarse pares de registros. Los pares son los siguientes: bc, de, y hl.

Es importante recalcar que se modifica el contenido del par 'de' tanto si se modifica d como si se modifica e. Sin embargo en BASIC la variable 'de' es totalmente independiente de las variables 'd' y 'e'. Este problema desaparece pronto en cuanto nos acostumbramos al lenguaje. Otra limitación consiste en el número de operaciones que pueden hacerse con estas variables. La mayor parte de la actividad debe realizarse en el registro 'a' (acumulador). Cuando se necesitan realizar operaciones aritméticas con números demasiado grandes para 'a', se utiliza el par de registros 'hl'. Los registros 'a' y 'hl' son mucho más versátiles que los demás.

Para trabajar en código máquina es necesario un cambio de procedimiento. Tiene que dividirse el trabajo en pasos cortos y simples. Los datos tienen que extraerse de su almacenamiento, someterse a manipulación y volver a colocarlos en su sitio antes de continuar con el siguiente paso. Este procedimiento no es inflexible pero es mucho más útil en lenguaje máquina que en BASIC. Frecuentemente se necesita almacenar datos en lugares específicos cuando se trabaja en BASIC. Esto al principio resulta incómodo. También nos puede dar la impresión que la ejecución del programa será lenta debido al número de instrucciones. Se diría que los programas como éstos escritos en BASIC deberían ejecutarse mediante la orden WALK (camina) en lugar de mediante RUN (corre). El código máquina, sin embargo, parece tener la velocidad de la luz al compararlo con el BASIC y esto es justo lo que se necesita para las facilidades que tenemos que añadir para convertir SPECPRO en SPECTEXT.

En las páginas que siguen y en el Capítulo 9, se listan seis programas en lenguaje máquina después de los programas BASIC de los que son traducción. No es preciso introducir en la computadora los programas BASIC, se listan tan solo para poner de relieve como se han desarrollado los otros. Se documenta el BASIC en vez del código máquina porque el lector se sentirá probable-

mente más a gusto con el primero. La tabla 6.1 contiene las 34 instrucciones máquina que se precisan para escribir estos seis programas. En la segunda columna de la tabla aparecen los equivalentes BASIC. Gran parte de las instrucciones son fáciles de comprender pero algunas de ellas necesitan explicación. El registro hl son en realidad dos registros en uno. Cada registro simple puede contener un número comprendido entre Ø y 255 al igual que dos posiciones de memoria. El par de registros hl necesita dos posiciones de memoria para volcar su contenido.

Las instrucciones del lenguaje máquina son más simples de lo que puede parecer. Son nemónicos, nombres cortos que sugieren la acción que realizan. Por ejemplo, la instrucción 'ld' significa carga (load en inglés) y 'jp' significa salta (jump en inglés). Hay variantes de la instrucción 'jp': 'jpz' significa salta si el último cálculo ha dado como resultado cero, 'jpm' realiza el salto si el resultado del último cálculo fue negativo y 'jpp' en caso de resultado positivo. La instrucción 'call' llama a una subrutina y 'ret' retorna de la misma. Cada programa necesita al menos una instrucción 'ret' para volver al BASIC; 'call' y 'ret' tienen versiones en las que verifican el resultado del último cálculo antes de decidir si cumplen o no su cometido. En la versión BASIC de las instrucciones, se hace referencia al registro 'a' para simplificar las cosas, pero el último cálculo podría haberse realizado en el registro 'hl' en cuyo caso, debería verificarse el contenido de este último. Instrucciones tales como que 'jpz' genera salto si el resultado del último cálulo realizado fue cero o 'ret nz' que retorna de una subrutina precisamente si el resultado del último cálculo no fue cero, son de uso frecuente en las rutinas. Es muy eficiente ejecutar 'jpz' o 'ret nz' inmediatamente después del cálculo que es la base de la decisión.

Los equivalentes en BASIC de las instrucciones 'push' y 'pop' disponen de una pequeña pila de números. La pila contiene una lista de números que manipula con un método LIFO (último en entrar primero en salir), igual que la pila real de la computadora. El puntero de la pila se incrementa una unidad en cada operación con la pila. El puntero de la pila real se incrementa en dos, pero el efecto es el mismo.

Las dos instrucciones que faltan por explicar son 'lddr' y 'ldir', dos de las más potentes del conjunto de instrucciones del Z80. En ambos casos se trasladan bloques de información desde un lugar a otro. En ambos casos el par de registros 'de' contiene la dirección de destino, 'hl' la dirección de origen y 'bc' el número de bytes que deben moverse. La instrucción 'ldir' traslada desde hl hasta de, suma uno a ambos y resta uno de bc, repite este proceso hasta que bc alcanza el valor cero. Esta instrucción se conoce como

'carga con incremento y repetición'. La segunda operación es muy similar pero en lugar de sumar, se resta uno de hl y de. La instrucción 'lddr' se conoce como (carga con decremento y repetición). La necesidad de los dos métodos de transferencia de información se explica más adelante.

Sorprendentemente, no es preciso un programa muy largo para llevar a cabo los objetivos que nos hemos propuesto. Necesitaremos un par de rutinas para mover bloques que se basarán en 'ldir' y 'lddr' y una rutina para rellenar un área de memoria con caracteres espacio. La única rutina complicada que necesitaremos será la correspondiente a la localización de un conjunto de caracteres dentro del texto. Empezaremos con la más simple de todas, la que rellena bloques de memoria con espacios. Esta rutina es necesaria porque la memoria de la máquina contiene ceros si no se ha introducido nada y éstos se imprimen como '?'. Si rellenamos un bloque de memoria con el carácter 32 (espacio) parecerá que estamos escribiendo sobre papel en blanco como es usual. Aquí se presenta la rutina BASIC que realiza este trabajo. No es preciso introducir el programa en BASIC, es el programa en código máquina que sigue lo que nos interesa.

```
  1 REM BASIC version of FILL,
        for demonstration only,
        not to be entered
 1Ø LET m1=32
 2Ø LET m2=start address
 3Ø LET m3=end address
 4Ø DIM z(1ØØ): LET sp=1
 5Ø LET hl=m2
 6Ø LET bc=m3
 7Ø LET a=m1
 8Ø POKE hl,a
 9Ø LET hl=hl+1
1ØØ LET z(sp)=hl: LET sp=sp+1
11Ø IF hl=Ø THEN  GO TO 14Ø
12Ø LET sp=sp-1: LET hl=z(sp)
13Ø GO TO 8Ø
14Ø LET sp=sp-1: LET hl=z(sp)
15Ø RETURN
```

En código máquina no se tienen que usar números de línea pero debe introducirse en una posición concreta de memoria y a partir de ella se ejecuta secuencialmente. La dirección de comienzo de esta rutina será 6Ø 151 y si se dispone de un programa

ensamblador como el de McGraw-Hill/ZX Spectrum debe usarse este número como origen. En caso de no disponer de ensamblador no hay que preocuparse, al final de este capítulo se da un método para introducir el programa sin utilizarlo. Un ensamblador es un programa que trasforma los nemónicos que pueden ser «entendidos» por las personas en números que puede entender la computadora. Aquí se tienen los nemónicos o códigos de operación correspondientes al anterior programa BASIC.

```
  1 REM FILL mnemonics to be
        assembled
 1Ø REM go
 2Ø REM org 5ØØØØ 6Ø151
 3Ø REM ld hl,Ø
 4Ø REM ld bc,Ø
 5Ø REM ld a,Ø
 55 REM B
 6Ø REM ld (hl),a
 7Ø REM inc hl
 8Ø REM push hl
 9Ø REM sbc hl,bc
1ØØ REM jp z,A
11Ø REM pop hl
12Ø REM jp B
125 REM A
13Ø REM pop hl
14Ø REM ret
15Ø REM finish
16Ø RANDOMIZE USR 58ØØØ
```

El programa en código máquina parece más pequeño que el escrito en BASIC. De hecho el espacio que ocupa es despreciable en comparación con el correspondiente al BASIC. Este programa ocupa 21 bytes mientras que el mismo escrito en BASIC ocupa 250 bytes.

Los nemónicos del código máquina se presentan del mismo modo a como se utilizan en el ensamblador de McGraw-Hill. La instrucción 'org' se encontrará en todos los listados para el ensamblador y significa origen, es decir, la dirección del primer byte de la memoria que será ocupado por el programa en código máquina. El ensamblador ocupa un área de memoria tal que las rutinas se ensamblan desde la dirección 5Ø ØØØ y se colocarán en su posición definitiva más tarde.

Cuando se ensambla una rutina mediante el ensamblador de

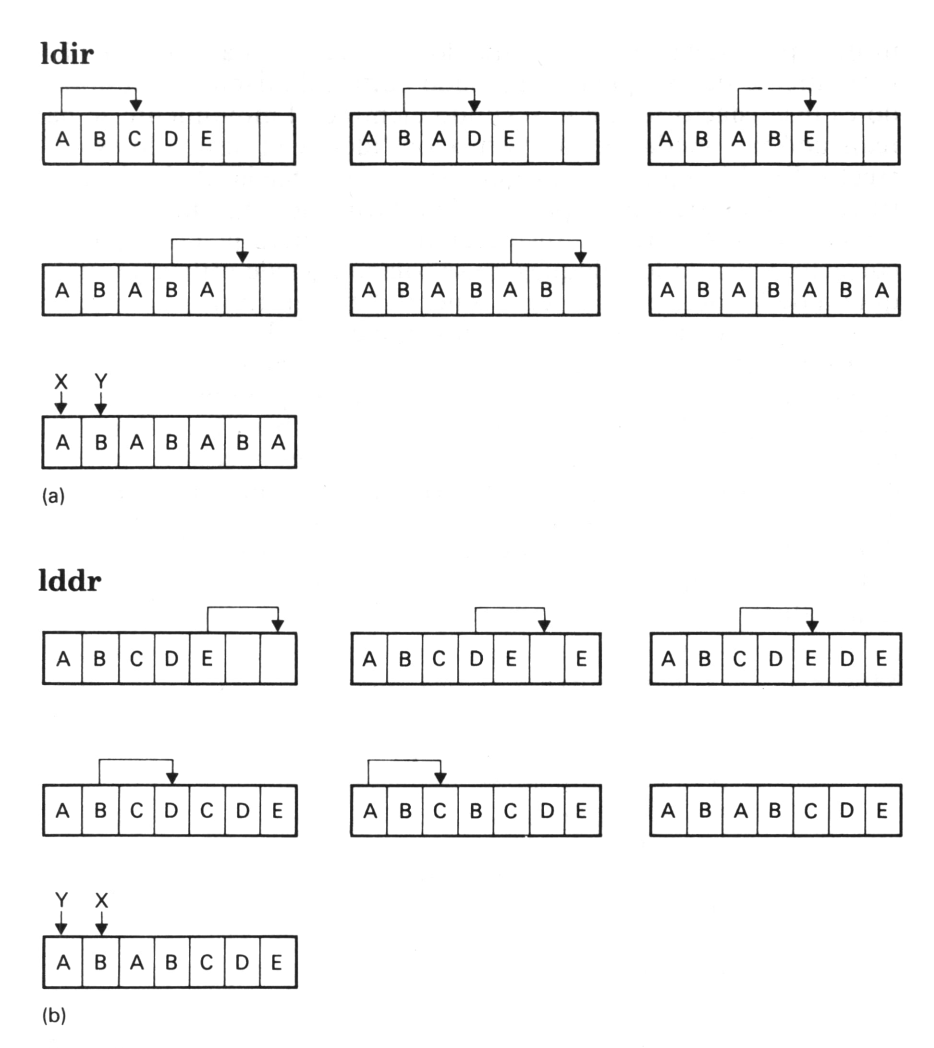


Figura 6.1

McGraw-Hill, debe grabarse en cinta por medio del siguiente comando:

SAVE “FILL” CODE 50000, 22

Si se utiliza otro ensamblador, examinar el manual de instrucciones.

Ahora que se ha dado a luz la primera rutina sin que haya sido demasiado doloroso el parto, podemos dedicarnos a las dos rutinas que mueven bloques de memoria. En primer lugar ¿por qué dos? En la figura 6.1 se explica. En el primer ejemplo ‘ldir’ se usa para mover un bloque de cinco caracteres hacia arriba en la me-

moria, para dejar espacio para dos caracteres más, X e Y. El movimiento del bloque ha cambiado parte del mismo y el resultado es insatisfactorio. En el segundo ejemplo el movimiento se ha realizado por medio de la instrucción 'lddr' y el resultado es correcto. Si el bloque tuviera que ser movido hacia abajo, la instrucción 'lddr' destruiría parte de la información durante el movimiento y 'ldir' sería la instrucción que generaría el resultado correcto. Cuando los bloques no se solapan puede utilizarse cualquiera de las dos, pero la regla general debe ser utilizar 'ldir' para mover hacia abajo y 'lddr' para mover hacia arriba.

El programa BASIC desde el que puede pasar a código máquina es, quizás, un poco más complicado de lo que podría ser. Las líneas comprendidas entre la 8Ø y la 1ØØ cambian los valores de los 'registros' como lo haría 'ldir' aunque no es necesario, puesto que, la computadora retorna en seguida al programa principal. No obstante, si los valores de los registros van a usarse después de 'ldir' o 'lddr', el programa BASIC deberá incluir líneas como las anteriormente mencionadas, de lo contrario, la simulación no será realista.

Move

```
  1 REM BASIC version of MOVE,
        demonstration only, not
        to be entered
 1Ø LET bc=number of bytes
 2Ø LET hl=destination
 3Ø LET de=hl
 4Ø LET hl=start address
 5Ø FOR j=Ø TO bc-1
 6Ø POKE de+j,PEEK hl+j
 7Ø NEXT j
 8Ø LET hl=hl+bc
 9Ø LET de=de+bc
1ØØ LET bc=Ø
11Ø RETURN
```

El código máquina correspondiente a este programa BASIC es también muy corto, ocupa tan solo 28 bytes.

Move

```
  1 REM MOVE  mnemonics to be
      assembled
 1Ø REM go
 2Ø REM org 5ØØØØ 6ØØØØ
 3Ø REM ld bc,Ø
 4Ø REM ld hl,Ø
```

```
 50 REM ld d,h
 60 REM ld e,l
 70 REM ld hl,0
 80 REM ldir
 90 REM ret
100 REM ld bc,0
110 REM ld hl,0
120 REM ld d,h
130 REM ld e,l
140 REM ld hl,0
150 REM lddr
160 REM ret
170 REM finish
180 RANDOMIZE USR 58000
```

Esta rutina debe grabarse en cinta inmediatamente después de la rutina FILL utilizando el comando.

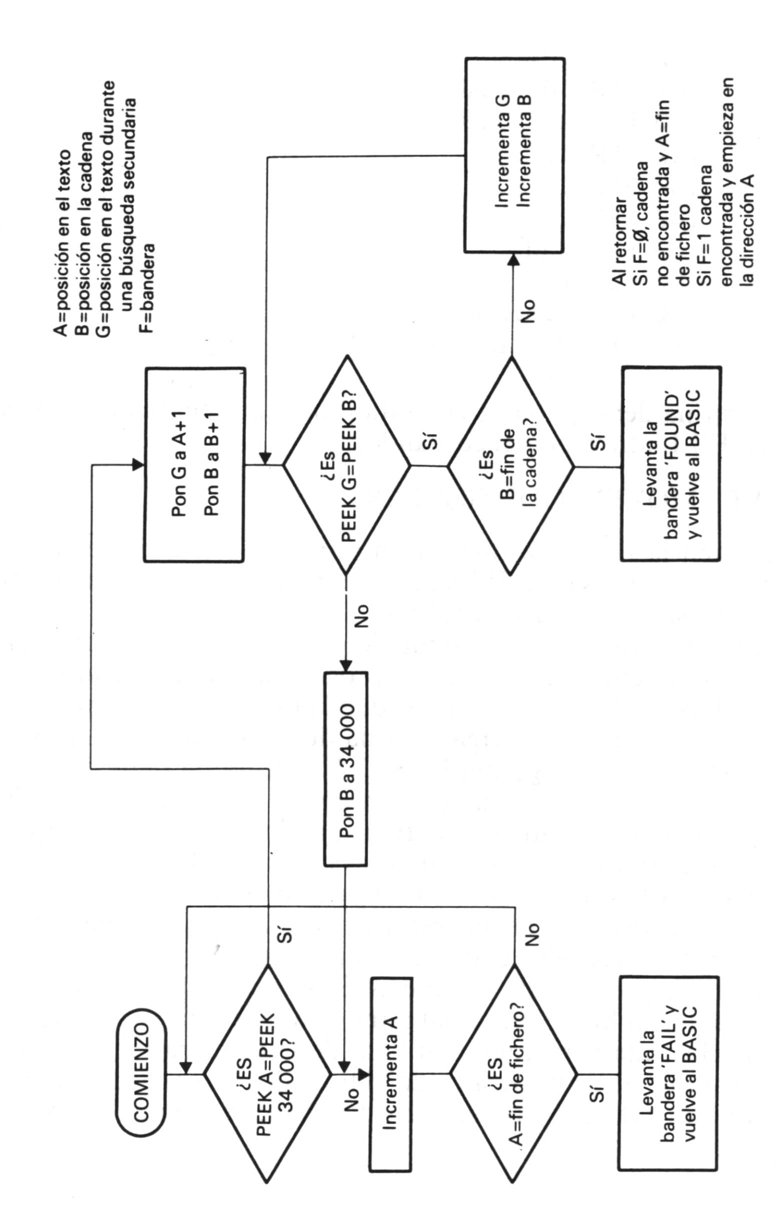
SAVE "MOVE"CODE 50000, 28

Los ceros en la rutina se cambiarán por sus valores correctos en cada movimiento usando POKE. En el Capítulo 4 se encuentran los detalles para inicializar la rutina.

Ahora que nos hemos introducido en el lenguaje máquina, ha llegado el momento de realizar un programa más avanzado, una rutina de búsqueda. La tarea de búsqueda es bastante complicada, por eso, se dispone de un organigrama del proceso (Fig. 6.2) para ver como trabaja el programa BASIC. El programa tiene dos partes. Hay un bucle principal que examina cada ocurrencia del primer carácter de la cadena que tiene que localizarse y un segundo bucle que comprueba los siguientes caracteres hasta el final de la cadena o hasta que se encuentra un carácter erróneo. Si se llega al final de la cadena sin encontrar un error, la búsqueda ha tenido éxito y la rutina devuelve el control al programa principal con detalles sobre la posición de la cadena. Si se encuentra un error, el control vuelve al bucle principal para localizar otra vez el primer carácter. Hay una verificación al final de la primera sección para comprobar que no se ha alcanzado el final del fichero (texto). Si se hubiera alcanzado el final del fichero la búsqueda habría sido infructuosa.

Search

El programa que sigue asustará a cualquiera que esté acostumbrado a programar optimizando la eficiencia en BASIC. No parece natural dividir el problema en partes tan pequeñas; a esto



A=posición en el texto
B=posición en la cadena
G=posición en el texto durante una búsqueda secundaria
F=bandera
COMIENZO
¿ES PEEK A=PEEK 34 000?
Sí
No
Pon G a A+1
Pon B a B+1
¿Es PEEK G=PEEK B?
Sí
No
Incrementa G
Incrementa B
¿Es B=fin de la cadena?
Sí
No
Levanta la bandera 'FOUND' y vuelve al BASIC
Pon B a 34 000
Incrementa A
¿ES A=fin de fichero?
Sí
No
Levanta la bandera 'FAIL' y vuelve al BASIC
Al retornar
Si F=Ø, cadena no encontrada y A=fin de fichero
Si F=1 cadena encontrada y empieza en la dirección A

Figura 6.2

nos referíamos antes cuando decíamos que se necesitaría una nueva actitud ante la programación. Aparte de las variables que representan a los registros, se utilizan otras seis variables, de m1 a m6. Estas variables representan seis pares de bytes de la memoria que se utilizan en la rutina de búsqueda. El primer paso del programa consiste en inicializar una de estas variables con la dirección del principio de la cadena almacenada a partir de la posición 34 000. El texto almacenado a partir de la dirección 35 000.

```
 1 REM BASIC version of FIND,
        for demonstration only,
        not to be entered
10 LET hl=34000
20 POKE m2,hl
```

La variable 'm1' se inicializa mediante POKE con la posición desde la que comenzar la búsqueda. Este valor se introduce en 'hl' y se incrementa antes de volver a colocarse en su sitio.

```
30 LET hl=PEEK m1
40 LET hl=hl+1
50 POKE m1,hl
```

En el siguiente paso, se carga 'bc' con la dirección del final del fichero que se almacena en 'm3'. La dirección contenida en 'hl' se compara con 'bc', si los contenidos son iguales se ha alcanzado el final del fichero. La computadora volverá al programa principal en este caso.

```
60 LET bc=PEEK m3
70 LET hl=hl-bc
80 IF hl=0 THEN  GO TO 390
```

Ahora se compara el primer carácter de la cadena con el correspondiente del texto.

```
 90 LET bc=PEEK m1
100 LET a=PEEK bc
110 LET hl=PEEK m2
120 LET L=PEEK hl
130 LET a=a-L
140 IF a=0 THEN  GO TO 160
150 GO TO 30
```

La siguiente sección se alcanzará únicamente cuando se haya alcanzado en el texto un carácter idéntico al inicial de la cadena y por tanto se tengan que comparar los restantes caracteres. Un solo fallo será suficiente para enviar a la computadora a proseguir la búsqueda desde donde se había detenido. Como tiene que continuarse desde donde se abandonó, debe preservarse el valor de 'm1'. El primer paso en la segunda parte del programa será adecuar el contador secundario para que contenga la dirección del siguiente carácter del texto.

```
160 LET hl=PEEK m1
170 LET hl=hl+1
180 POKE m6,hl
```

Ahora es preciso movernos dentro de la cadena para obtener el siguiente carácter de la misma y después se necesita una comprobación para averiguar si se ha llegado al final de la búsqueda. La dirección del final de la cadena está contenida en 'm4' y tiene que compararse con el contenido del registro 'hl'.

```
190 LET hl=PEEK m2
200 LET hl=hl+1
210 POKE m2,hl
220 LET bc=PEEK m4
230 LET hl=hl-bc
240 IF hl=0 THEN  GO TO 360
```

Los siguientes caracteres tanto de la cadena como del texto pueden compararse ahora. Si no coinciden, la línea 310 envía a la computadora a la primera parte de la rutina.

```
250 LET bc=PEEK m6
260 LET a=PEEK bc
270 LET hl=PEEK m2
280 LET L=PEEK hl
290 LET a=a-L
300 IF a=0 THEN  GO TO 320
310 GO TO 10
```

Antes de completar el segundo bucle, tiene que incrementarse el contador secundario para apuntar al siguiente carácter.

```
320 LET hl=PEEK m6
330 LET hl=hl+1
```

```
340 POKE m6,hl
350 GO TO 190
```

El control pasa a la línea 360 o a la línea 390 dependiendo del éxito de la búsqueda. La bandera 'm5' se pone a cero para indicar que no se ha encontrado la cadena dentro del texto. Si 'm5' tiene el valor 1, la dirección del primer carácter de la cadena dentro del texto está contenida en 'm1'.

```
360 LET a=1
370 POKE m5,a
380 GO TO 410
390 LET a=0
400 POKE m5,a
```

Antes de que la computadora vuelva al programa principal, la dirección del primer carácter de la cadena vuelve a tomar el valor 34 000.

```
410 LET hl=34000
420 POKE m2,hl
430 RETURN
```

Aquí se tienen los nemónicos del código máquina.

```
  1 REM FIND mnemonics to be
        assembled
 10 REM go
 20 REM org 50000 60028
 25 REM F
 30 REM ld hl,34000
 40 REM ld (60137),hl
 45 REM C
 50 REM ld hl,(60135)
 60 REM inc hl
 70 REM ld (60135),hl
 80 REM ld bc,(60139)
 90 REM sbc hl,bc
100 REM jp z,A
110 REM ld bc,(60135)
120 REM ld a,(bc)
130 REM ld hl,(60137)
140 REM ld l,(hl)
```

```
15Ø REM sub l
16Ø REM jp z,B
17Ø REM jp C
175 REM B
18Ø REM ld hl,(6Ø135)
19Ø REM inc hl
2ØØ REM ld (6Ø145),hl
2Ø5 REM G
21Ø REM ld hl,(6Ø137)
22Ø REM inc hl
23Ø REM ld (6Ø137),hl
24Ø REM ld bc,(6Ø141)
25Ø REM sbc hl,bc
26Ø REM jp z,D
27Ø REM ld bc,(6Ø145)
28Ø REM ld a,(bc)
29Ø REM ld hl,(6Ø137)
3ØØ REM ld l,(hl)
31Ø REM sub 1
32Ø REM jp z,E
33Ø REM jp F
335 REM E
34Ø REM ld hl,(6Ø145)
35Ø REM inc hl
36Ø REM ld (6Ø145),hl
37Ø REM jp G
375 REM D
38Ø REM ld a,1
39Ø REM ld (6Ø143),a
4ØØ REM jp H
4Ø5 REM A
41Ø REM ld a,Ø
42Ø REM ld (6Ø143),a
425 REM H
43Ø REM ld hl,34ØØØ
44Ø REM ld (6Ø137),hl
45Ø REM ret
46Ø REM finish
47Ø RANDOMIZE USR 58ØØØ
```

Find

Esta rutina debe grabarse en cinta como se ha hecho con sus antecesoras FILL y MOVE. El comando que tiene que utilizarse es:

SAVE "FIND"CODE 50000, 107

Ahora pueden volver a grabarse las tres rutinas juntas y listas para trabajar. Rebobina la cinta y carga todas las rutinas en sus posiciones correctas por medio de los siguientes comandos:

LOAD "FILL"CODE 60151 LOAD "MOVE"CODE 60000
LOAD "FIND"CODE 60028

El siguiente paso es grabar todas las rutinas conjuntamente mediante el comando:

SAVE "mc"CODE 60000, 175

Ahora puede almacenarse en cinta todo el paquete SPEC-TEXT. Las siguientes líneas constituyen un programa de carga que realiza algunas funciones y prepara después la carga del resto del paquete. El bucle FOR... NEXT es para rediseñar el primer gráfico definido por el usuario, CHR$ 144, para las dobles comillas. La instrucción CLEAR protege el texto y el código máquina de la acción del comando NEW.

```
  10 CLEAR 33999: FOR j=0 TO 7:
POKE USR "a"+j,0: NEXT j: POKE U
SR "a"+1,36: POKE USR "a"+2,36
  20 PRINT  FLASH 1;AT 4,7;" SPE
CTEXT LOADING ": PLOT 50,132: DR
AW 154,0: DRAW 0,15: DRAW -154,0
: DRAW 0,-15
  30 PRINT ''"Two more parts to
load": LOAD "mc"CODE : PRINT "On
e more part to load": LOAD "word
"
```

Grabe este programa en una cinta virgen por medio del comando:

SAVE "spectext" LINE 1

después extraiga la cinta del magnetófono sin rebobinarla. Tiene que añadirse el resto de SPECTEXT inmediatamente después de este pequeño programa. Cargue el código máquina en la computadora con:

LOAD "mc"CODE

ahora se añade a continuación del programa cargado con:

SAVE "mc"CODE 60000, 175

El paquete se completa cargando la parte del programa escrita

en BASIC en la computadora, para grabarla después a continuación de las otras dos partes de SPECTEXT con el comando:

SAVE "word"LINE 60000

Debemos tener copias en cinta de todas las partes y la versión total. El paquete puede probarse tecleando LOAD " " y ENTER y conectando la cinta. El programa debe comenzar con el menú de opciones nada más ser cargado. La información para manejar todas las funciones está en el Capítulo 5.

Aquellos lectores que no dispongan de un ensamblador tendrán que introducir las rutinas en código máquina por algún otro procedimiento. Afortunadamente hay otro método que no es difícil. El código máquina no es sino una serie de números. Los nemónicos son un puro convenio para hacer el programa inteligible. Los números pueden ser almacenados en memoria por medio de la instrucción POKE. Una forma de almacenar los números del código en la memoria es por medio de un bucle FOR... NEXT y de una instrucción DATA. Es absolutamente imprescindible que los números se introduzcan correctamente. Una pequeña equivocación en un número podría ser suficiente para estropear todo el programa. Haga copias de todos los programas BASIC al introducirlos. Si hubiera errores podrá corregirlos sólo si se dispone de una copia para editar.

Aquí se tiene el primer bloque de números correspondientes al código máquina de la rutina MOVE.

```
Numeric code for MOVE

1    0    0    33   0    0
84   93   33   0    0    237
176  201  1    0    0    33
0    0    84   93   33   0
0    237  184  201
```

Move

La sentencia DATA correspondiente a este bloque es:

```
10 DATA 1, 0, 0, 33, 0, 0, 84, 93, 33, 0, 0, 237, 176, 201,
1, 0, 0, 33, 0, 0, 84, 93, 33, 0, 0, 237, 184, 201
20 FOR j=60000 to 60027: READ a: POKE j,a: NEXT j
```

Con la línea 20 en su sitio puede ejecutarse el programa mediante RUN y así el código se escribirá en sus posiciones correctas. Escriba un programa similar para los siguientes bloques de números, pero primero grabe esta parte del código máquina con el comando:

SAVE “MOVE”CODE 60000, 28

Cuando se construyan las otras rutinas será conveniente dividir los largos bloques de números en varias líneas DATA, puesto que, si se produce un error es más sencillo editar una línea corta que una larga. El bucle FOR... NEXT de la línea 20 debe editarse para que los extremos tomen sus nuevos valores que son los que aparecen en la parte superior de cada bloque.

```
Numeric code for FILL
FOR j=60151 TO 60172

33   0    0    1    0    0
62   0    119  35   229  237
66   202  11   235  225  195
255  234  225  201
```

La rutina FILL debe grabarse por medio del comando:
SAVE “FILL”CODE 60151, 22

```
Numeric code for FIND
FOR j=60028 TO 60134

33   208  132  34   233  234
42   231  234  35   34   231
234  237  75   235  234  237
66   202  219  234  237  75
231  234  10   42   233  234
110  149  202  162  234  195
130  234  42   231  234  35
34   241  234  42   233  234
35   34   233  234  237  75
237  234  237  66   202  211
234  237  75   241  234  10
42   233  234  110  149  202
201  234  195  124  234  42
241  234  35   34   241  234
195  169  234  62   1    50
239  234  195  224  234  62
0    50   239  234  33   208
132  34   233  234  201
```

Find

La rutina FIND debe grabarse mediante el comando:
SAVE “FIND”CODE 60028, 107

Cargue después los tres programas y salve las tres rutinas conjuntamente con el comando:

SAVE "mc"CODE 60000, 175

Después continúe con las instrucciones de la página 75.

Este capítulo así como el Capítulo 9 podrían estimular el interés en los lenguajes de bajo nivel al poner de manifiesto las semejanzas entre el código máquina del microprocesador Z80 y el BASIC. Hay muchas más instrucciones y técnicas además de las que hemos manejado aquí. La potencia de las rutinas que hemos desarrollado podría tentar a los lectores a ampliar sus conocimientos en esta área, en este caso les serán de utilidad los siguientes libros:

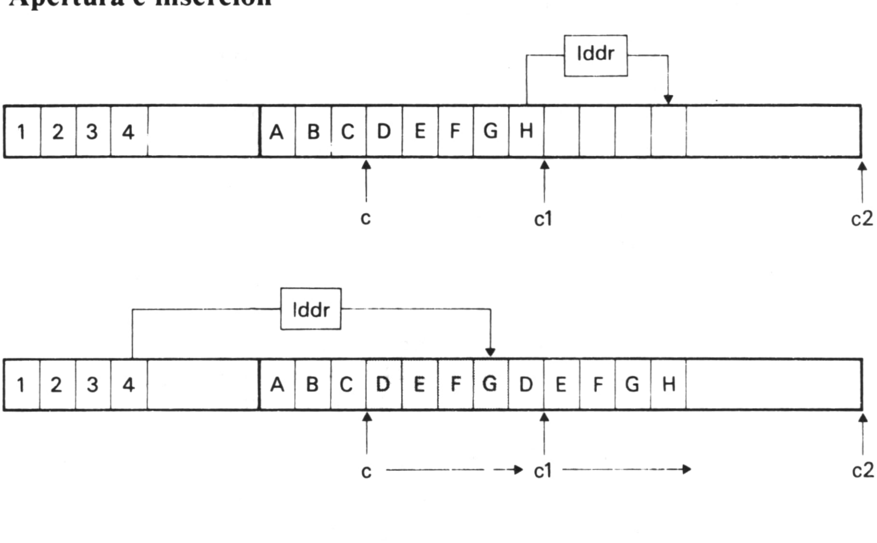
Woods, Tony, *Learn and Use Assembly Language on the ZX Spectrum,* 1983.

Street, C. A., *Information Handling for the ZX Spectrum,* 1983.

Gosling, Peter, *Structured Programing—A First Course for Students and Hobbyists,* 1983.

Bishop, Graham, *Spectrum Interfacing and Projects,* 1983.

Apertura e inserción



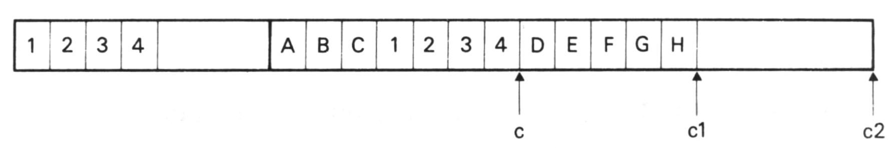


Figura 6.3

Borrado y acercamiento

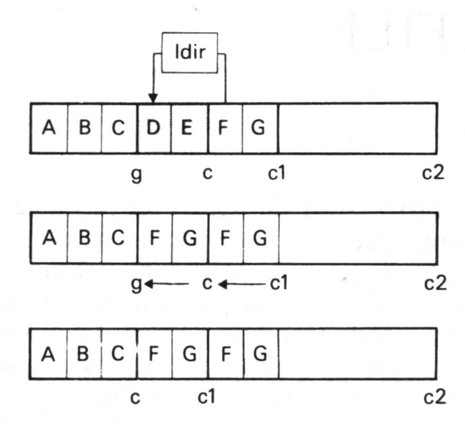


Figura 6.4

Alteración

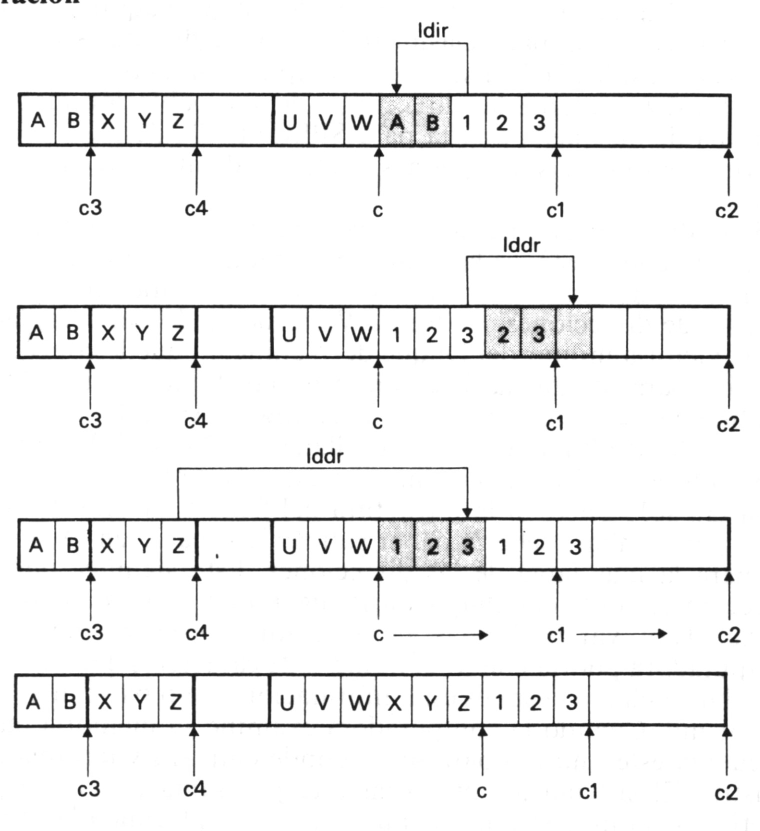


Figura 6.5

7 SPECFILE

Finalizado el procesador de textos, el paso siguiente es desarrollar un programa que permita el envío de cartas a los componentes de una lista de correo. Dicho programa va a obtenerse como fusión de otros dos, uno de los cuales permitirá la escritura de un texto estandarizado (modelo de carta) y el segundo adecuará el mismo para su envío a algunos o todos los componentes de una lista (fichero). Es necesario que el programa sea más flexible que el anterior. ¿Por qué limitarse a una lista de personas?, ¿por qué no escribir un programa que permita su utilización en trabajos diferentes? Para la construcción de un programa como éste se necesitan nuevas rutinas escritas en lenguaje máquina, una de ellas deberá servir para clasificar la lista en diferentes órdenes. Una vez clasificada la lista, por ejemplo, por poblaciones, podrían localizarse todas las direcciones de una población mediante la rutina de búsqueda escrita para SPECTEXT, después podrían imprimirse todas las direcciones correspondientes a dicha población.

Una decisión que ha de tomarse pronto cuando se diseña un programa como éste, es la forma de almacenamiento de la información. Para comprender el problema nos bastará con una simple lista de direcciones. La Reina Victoria quedó tan complacida por la victoria militar del Duque de Wellington que le concedió el permiso para utilizar la dirección especial: Londres núm. 1. Generalmente, las direcciones son más largas: Avenida de Burgos, 29, 4-d, Madrid; Paseo de la Castellana, 81, Madrid. Esta última dirección tiene 33 caracteres mientras que la dirección del Duque tenía tan solo once. Si la estructura del fichero tuviera que diseñarse para ambas, debería haber suficiente espacio en cada registro para la más larga de las direcciones. Esto significa que una dirección como la del Duque daría lugar a un desperdicio de memoria. Hay varias formas de solucionar este problema. La computadora podría contar el número de caracteres de cada registro, almacenando después esta información como parte integrante del mismo. Cuando la computadora examine su memoria tendrá en cuenta este número para saber donde empieza y termina cada registro. El sistema adoptado en este programa es simple pero efectivo. Con un conjunto de marcadores se informa a la computadora su posición en cada momento. Se utilizan los mismos mar-

cadores que en el procesador de textos; como consecuencia, pueden usarse las mismas rutinas.

El fichero de información se divide así en unidades más pequeñas mediante la utilización de estos marcadores. El marcador que se utilizaba en el procesador de textos para indicar el final de una página, sirve aquí para señalar el final del fichero. Cada registro se separa de los demás mediante el mismo carácter usado por el procesador de textos para indicar el final de un párrafo. Los campos se separan mediante el marcador de 'nueva línea' de SPECTEXT. Usando este sistema, en teoría, cada registro podría tener cualquier longitud. Sin embargo, es conveniente moderar la longitud de cada campo, puesto que el registro debe caber en la pantalla. Se permiten dos tipos de campos, uno corto que aparezca junto a su rótulo en una sola línea y otro largo que puede llegar a ocupar dos líneas de la pantalla. Cada registro puede contener hasta 15 campos.

SPECTEXT y SPECFILE son programas con una estructura muy similar. Cuanto más se parezcan, más fácil será integrarlos para formar otro nuevo programa más potente. SPECFILE está escrito en BASIC utilizándose el lenguaje máquina solo cuando el BASIC no es suficientemente rápido. El propio programa junto con la documentación y comentarios que se acompañan al listado constituyen suficiente fuente de información como para que algunos usuarios realicen modificaciones y adapten el programa a sus propias necesidades. Uno de los capítulos está expresamente dedicado a explicar el código máquina. Es fundamental que estas rutinas escritas en lenguaje máquina sean lo suficientemente generales como para que puedan utilizarse en otros programas del usuario.

SPECFILE dispone de un menú principal con diversas opciones. Hay otros menús secundarios diseñados de tal forma que no interfieren la imagen de la pantalla. Otra característica del programa es la utilización de señales acústicas. Mediante un breve sonido se indica al usuario el final de cada proceso para que de este modo tome la decisión correspondiente. Generalmente lo que debe realizar el usuario después del sonido es confirmar o rechazar una entrada de datos. La confirmación se acompaña de un sonido agudo, mientras que el rechazo produce una señal más grave. Es fácil quedar hipnotizado por la pantalla durante el proceso de entrada de datos y presionar automáticamente la tecla correspondiente a la confirmación. La señal sonora nos ofrece una segunda oportunidad y permite la corrección de errores. Se ha elegido una señal acústica diferente para confirmar la eliminación de un registro de un fichero, de este modo el borrado accidental es más difícil.

La instrucción RANDOMIZE, que ha sido utilizada anteriormente vuelve a usarse de nuevo más profusamente. Inicialmente, el programa era mucho más largo que en la actualidad, principalmente por el número de veces que era necesario utilizar la instrucción POKE para introducir una dirección en dos posiciones de memoria y la instrucción PEEK para leer una variable del código máquina que ocupase dos bytes. En algunas versiones del BASIC hay dos instrucciones y gran utilidad: DEEK y DOKE, que permiten efectuar dobles PEEK y POKE, respectivamente. En este programa se han preparado dos cortas subrutinas que nos permiten disponer de estas funciones, ahorrando de este modo una gran cantidad de memoria. La desventaja es que el listado del programa contiene muchas instrucciones RANDOMIZE que le hace parecer más difícil de lo que realmente es. La presentación es la misma que en los capítulos anteriores, por tanto no habrá dificultad en seguir el programa.

Los registros construidos con este programa pueden tener hasta 536 caracteres. El buffer de línea está situado entre las direcciones 34 ØØØ y 34 999 y contiene un registro completo, así como todos los detalles del fichero: nombre, longitudes, etc. La rutina de clasificación necesita un espacio para el almacenamiento temporal de los registros, por eso utiliza la zona de memoria situada por encima del código máquina. En el capítulo dedicado al código máquina se explica como se puede modificar la rutina de clasificación para que la información se almacene temporalmente allí donde haya espacio disponible. De este modo el software será trasladable para su uso en otros programas. La rutina de clasificación en lenguaje máquina es la más larga de este libro y sería una lástima que todos los esfuerzos fueran hacia una sola implementación.

EL LISTADO DEL PROGRAMA

El programa comienza con una línea que salta todas las subrutinas enviando el control a la línea 9ØØ que es donde se escribe el menú principal. Las subrutinas tendrán más sentido cuando se haya examinado el resto del programa, por tanto, el primer paso es el menú. La composición es la misma que en ocasiones anteriores con siete opciones numeradas desde la 1 a la 7. Mediante la línea 91Ø se verifica que la opción elegida está comprendida entre 1 y 7. El menú borra la pantalla dos veces, la primera para borrar lo anteriormente expuesto y la segunda para preparar la salida correspondiente a la opción seleccionada. Con esta sencilla técnica nos ahorramos multitud de borrados a lo largo del progra-

ma. La línea 920 dicta la forma del programa conformándolo en bloques de 1000 líneas.

```
 9ØØ CLS : PRINT TAB 9;" SPEC FI
LE "''''"1   Add to file"'"2   S
ort file"'"3   Search/Edit file"
'"4   Load file from tape"'"5
Record file"'"6   Design new fil
e"'"7   Stop work"
 91Ø LET a$=INKEY$: IF a$<"1" OR
 a$>"7" THEN  GO TO 91Ø
 92Ø CLS : GO TO 1ØØØ*VAL a$
```

La primera de estas secciones permite la entrada de datos en el fichero previamente creado mediante la sección de la línea 6000. Esta rutina se ejecuta solamente si se ha creado previamente un fichero, por tanto, la primera línea analiza el contenido de una de las posiciones de memoria rellenada por la opción que permite la creación de ficheros. Si el contenido de esta posición fuera cero, significaría que el fichero no existe y se retornaría al menú.

```
1ØØØ IF PEEK 34997=Ø THEN  GO TO
 1
```

Para ahorrar espacio, las subrutinas se indican mediante los números de su primera línea entre paréntesis. La rutina (100) rellena la larga cadena de la línea 2 con espacios y (50) imprime esta cadena para borrar parte de la pantalla. La variable 'c2' contendrá la dirección del final del buffer y la variable 'c' servirá de contador de caracteres. En todo momento se compara 'c' con 'c2' para asegurarse que queda espacio disponible. Si el buffer estuviera completo la línea 1500 lo evidenciaría. La rutina (40) imprime un formulario en blanco tal y como se ha diseñado y (45) añade las etiquetas a los campos del registro. La rutina (50) utiliza PRINT OVER 1 para unir todo y ofrecer al usuario un formulario que cumplimentar.

```
1Ø1Ø GO SUB 1ØØ: GO SUB 5Ø: IF c
>c2 THEN  GO TO 15ØØ
1Ø2Ø GO SUB 4Ø: GO SUB 45: GO SU
B 55
```

En el caso de que una entrada sea rechazada debido a un

error, el valor del contador de caracteres 'c' se almacena en 'c4'. El bucle se prepara de acuerdo con el número de campos ('a') en el registro. Lo primero que hay que hacer es preparar el lugar donde aparecerá en la pantalla el primer campo. Las instrucciones PEEK examinan la información almacenada por la opción 6 en relación con la posición que cada campo adoptará en la pantalla. Cuando se localizan las coordenadas 'm' y 'n' la computadora imprime el cursor intermitente mostrando donde aparecerá el primer carácter. Si la información continúa en la siguiente línea, la bandera 'f' toma el valor 1. La variable 'b1' toma el valor de la longitud de cada campo. El número de columna 'b' puede moverse entre 'n' y 'b1'.

```
1Ø3Ø LET c4=c: FOR j=1 TO a: LET
 z=PEEK (34848+j*2)+256*PEEK (34
849+j*2)-23779+64: LET m=INT (z/
32): LET n=z-m*32+9
1Ø4Ø PRINT  FLASH 1;AT m-1,n;" "
;AT m-1,n;: LET b=n: LET f=Ø: LE
T b1=23+c: IF PEEK (34889+j)=2 T
HEN  LET b1=55+c
1Ø5Ø IF PEEK (34889+j)=3 THEN  L
ET b1=b1+32
```

La rutina (1Ø) lee el teclado y hace que la variable b$ tome el valor del carácter que se haya pulsado. Si el código de b$ no correspondiera a uno de los caracteres permitidos, la rutina de la línea 12ØØ efectuaría un ajuste. La línea 1Ø7Ø mantiene el texto dentro de los límites de longitud del campo y la 1Ø8Ø escribe en la pantalla los caracteres después de comprobar el valor de 'b'. Si esta variable alcanzase un valor superior a 31 caracteres se colocaría la bandera y se comenzaría una nueva línea. La línea 1Ø9Ø actualiza el buffer y la 11ØØ envía a la computadora en busca de una nueva pulsación.

```
1Ø6Ø GO SUB 1Ø: IF CODE b$>127 O
R CODE b$<32 THEN  GO TO 12ØØ
1Ø7Ø IF c=b1 THEN  GO TO 1Ø6Ø
1Ø8Ø PRINT b$;: LET b=b+1: IF b=
32 THEN  LET b=Ø: LET f=1: LET m
=m+1
1Ø9Ø POKE c,CODE b$: LET c=c+1
11ØØ GO TO 1Ø6Ø
```

Las siguientes líneas están desordenadas pero ayudan a la comprensión del tema. Si se hubiera pulsado la tecla ENTER (código 13), la línea 1200 enviaría el control del programa a otra sección. Esto significaría que se ha completado un campo. La línea 1210 rechaza cualquier pulsación a excepción de DELETE, e incluso esta última si se trata de borrar caracteres antes del comienzo del campo. Las líneas 1220 y 1230 actualizan las variables, eliminan de la pantalla y del buffer los caracteres no deseados y envían a la computadora en busca de otro carácter.

```
1200 IF CODE b$=13 THEN  GO TO 1
110
1210 IF CODE b$<>12 OR (b=n AND
f=0) THEN  GO TO 1060
1220 LET b=b-1: IF b<0 THEN  LET
 b=31: LET m=m-1: LET f=0
1230 PRINT AT m-1,b;" ";AT m-1,b
;: LET c=c-1: GO TO 1060
```

Ahora podemos revisar la sección que comienza en la línea 1110, que anteriormente fue pasada de largo. La computadora alcanza este punto, cuando el usuario después de completar un campo, pulsa la tecla ENTER. La línea 1110 produce un sonido agudo y hace después algo que puede parecer extraño. Lee el carácter impreso en m, n volviendo a escribirlo exactamente en el mismo lugar. Hay una buena razón para ello. Si el usuario decidiera dejar vacío el campo, el cursor intermitente tendría que borrarse y la instrucción 'PRINT AT m, n;' reemplazaría a la anterior 'PRINT FLASH 1; AT m, n;'. El resultado es que el cursor desaparece. Se añade un espacio (CHR$ 32) al registro, a continuación viene un marcador de nueva línea (CHR$ 131) que señala el final de un campo y, por último, un marcador de final de registro (CHR$ 139). En el caso de que no se hubiera llegado al final del registro, el último carácter sería sobre-impreso. El contador de caracteres 'c' se incrementa en dos posiciones y la computadora continuará con el bucle. Cuando se alcanza el último ciclo del bucle, se incrementa una pocisión el contador de caracteres, con objeto de no machacar el marcador de fin de registro. La línea 1140 produce un sonido para recordar al usuario que se ha llegado al final del registro y que es necesaria la confirmación antes de ser aceptado. La rutina (70) solicita al usuario que pulse la tecla 'C' en caso de estar de acuerdo ('Key C if OK'). 'Z' es la señal para rechazar el registro y volver de nuevo al menú. Pulsando únicamente ENTER se rechaza el registro, pero puede volver a introducirse. Ambas señales hacen que 'c' tome el mismo valor

que tenía al comienzo del registro y que está almacenado en 'c4'. Si se pulsa 'C' se incrementa el número de registro y se imprime un nuevo formulario mediante la rutina (45).

```
1110 BEEP .1,24: PRINT AT m-1,n;
SCREEN$ (m-1,n)
1120 POKE c,32: POKE c+1,131: PO
KE c+2,139: LET c=c+2
1130 NEXT j: LET c=c+1
1140 BEEP .1,0: GO SUB 70: IF b$
="z" THEN  LET c=c4: POKE c,143:
 GO TO 1
1150 IF b$<>"c" THEN  BEEP .1,-1
2: LET c=c4: GO SUB 45: GO TO 10
10
1160 BEEP .1,12: LET c5=c5+1: GO
 SUB 45: GO TO 1010
```

Las últimas líneas de esta sección son para indicar que el buffer está completo y el fichero tiene que grabarse en una cinta.

```
1500 PRINT  FLASH 1;AT 9,9;" BUF
FER FULL ": BEEP .1,20: IF INKEY
$="z" THEN  GO TO 1
1510 GO TO 1500
```

La sección siguiente es engañosamente corta. Estas siete líneas primero preparan y después ejecutan la rutina de clasificación escrita en lenguaje máquina que forma parte de este programa. Esta opción no puede usarse si no hay registros para clasificar, por eso la primera línea verifica que 'c' se ha desplazado de su posición inicial. La mayor parte del trabajo está en relación con la introducción de información en la zona de almacenamiento del código máquina.

```
2000 IF c=te THEN  GO TO 1
2010 CLS : RANDOMIZE te: POKE 60
173,PEEK q: POKE 60174,PEEK r
2020 LET z=0: POKE 60179,z: POKE
 60180,z: POKE 60183,z
```

La entrada de datos, como respuesta a la petición de la línea 2030, se verifica con (20) para asegurarse de que es un número

real y tiene el rango adecuado. Cuando se introduce un número apropiado, éste se facilita al código máquina. La línea 2050 almacena la información, llama a la rutina y genera los mensajes de error que fueran necesarios, cosa que debiera ocurrir muy raramente. Hay más información sobre los mensajes de error en el capítulo siguiente dedicado a las instrucciones operativas. La rutina (60) presenta el primer elemento de una lista clasificada y permite al usuario examinar tantos otros como desee.

```
2030 INPUT "Which Field?";z$: IF
 z$="" THEN  GO TO 1
2040 GO SUB 20: IF x=1 OR z=0 OR
 z>=a THEN  GO TO 2030
2050 POKE 60182,z: RANDOMIZE USR
 60185: IF PEEK 60183<>0 THEN  P
RINT ''"Error at ";PEEK 60173+25
6*PEEK 60174: INPUT z$: GO TO 20
10
2060 GO SUB 60: GO TO 2010
```

La opción de búsqueda es otra sección corta del programa que, sin embargo, realiza una gran cantidad de trabajo. La mayor parte está escrita en código máquina. Al igual que las anteriores, esta opción no puede utilizarse con un fichero vacío.

```
3000 IF c=te THEN  GO TO 1
```

La línea 3005 almacena el texto que ha de ser localizado en el buffer de línea (posición 34 000). El resto de la línea 3005, así como la línea 3010, utilizan la rutina (95) (rutina DOKE) para cargar información en el área destinada al código máquina y después obligan a éste a entrar en acción.

```
3005 PRINT " SEARCH ": INPUT "Ke
y text";a$: FOR j=1 TO LEN a$: P
OKE 33999+j,CODE a$(j): NEXT j:
RANDOMIZE te-1: LET d=60135: GO
SUB 95
3010 RANDOMIZE li: LET d=60137:
GO SUB 95: RANDOMIZE c: LET d=60
139: GO SUB 95: RANDOMIZE li+LEN
 a$: LET d=60141: GO SUB 95: POK
E 60143,0: RANDOMIZE USR 60028
```

Si la rutina FIND alcanzase el final del fichero sin haber encontrado el material especificado, se colocaría una bandera que después se verifica por medio de la línea 3Ø2Ø. Si se localizase una ocurrencia de la cadena, la rutina FIND señalaría su posición en el buffer. El programa necesita saber dónde comienza el registro, por este motivo la rutina (8Ø) recurre a otra rutina corta escrita en lenguaje máquina que sirve para localizar el principio y el final de cualquier registro.

```
3Ø2Ø IF PEEK 6Ø143=Ø THEN  PRINT
 " END OF FILE ": INPUT z$: GO T
O 1
3Ø3Ø GO SUB 8Ø
```

Al principio de este capítulo se hizo mención de las formas alternativas de presentar un menú de opciones sin perder la imagen existente en ese momento en la pantalla. Las siguientes cuatro líneas crean un menú de este tipo. Las opciones se imprimen en un pequeño receptáculo en la parte inferior de la pantalla. Pulsando ENTER se rechaza una opción y aparece impresa la siguiente. Cuando aparezca la opción deseada, bastará pulsar cualquier tecla distinta de ENTER para que ésta sea seleccionada.

```
3Ø4Ø INPUT "RUBOUT";a$: IF a$<>"
" THEN  GO TO 31ØØ
3Ø5Ø INPUT " COPY ";a$: IF a$<>"
" THEN  COPY
3Ø6Ø INPUT "ESCAPE";a$: IF a$<>"
" THEN  GO TO 1
3Ø7Ø GO TO 3Ø4Ø
```

La única opción que exige una acción amplia es la que sirve para borrar el registro que está en la pantalla. Las líneas que siguen hacen que sea difícil eliminar un registro accidentalmente. El código máquina que efectúa realmente el borrado es la parte más simple del programa y está controlado por la rutina (3Ø). Después del borrado, la línea 312Ø actualiza el contador de caracteres y el número de registro adecuándolos a la nueva situación. Finalmente se produce una confirmación acústica de que el trabajo ha concluido.

```
31ØØ CLS : PRINT  FLASH 1;AT 9,9
;"ARE YOU SURE?": INPUT z$: IF z
$<>"yes" THEN  GO TO 1
```

```
311Ø LET d=6Ø135: GO SUB 9Ø: LET
 v2=d: LET d=6Ø565: GO SUB 9Ø: L
ET v3=d+1: LET v1=c-d+1: GO SUB
3Ø
312Ø LET d=6Ø565: GO SUB 9Ø: LET
 z=d: LET d=6Ø135: GO SUB 9Ø: LE
T c=c-(z-d)-1: LET c5=c5-1: PRIN
T ''TAB 1Ø;" DELETED ": BEEP .3,
-2Ø: GO TO 1
```

Las dos secciones que siguen permiten grabar ficheros en cinta y cargar en la computadora ficheros previamente grabados para su actualización. Ambas secciones siguen el mismo sistema que el procesador de textos. Cuando se graba un fichero el nombre se toma de un espacio de memoria de 1Ø bytes destinado a tal fin. Mientras que se introduce el nombre del fichero se utiliza la rutina (7Ø) para permitir la corrección de errores y al nombre se le añaden los espacios necesarios para que la longitud del mismo sea 1Ø bytes.

```
4ØØØ CLS : PRINT " LOADING FILE
": INPUT "Key file name ";z$: IF
 z$="" THEN  GO TO 4ØØØ
4Ø1Ø PRINT '''z$: GO SUB 7Ø: IF
b$<>"c" THEN  GO TO 1
4Ø2Ø IF LEN z$<1Ø THEN  LET z$=z
$+" ": GO TO 4Ø2Ø
```

La información del fichero se graba secuencialmente y la instrucción de carga es LOAD CODE. Se graba también la información contenida en el buffer de línea, de este modo, la grabación contiene no solo los datos del fichero sino también su estructura, máscara, etc. Cuando se carga un fichero desde cinta, la línea 4Ø3Ø se encarga de almacenar toda esta información en las variables del programa.

```
4Ø3Ø LOAD z$CODE : LET c=PEEK 34
998+256*PEEK 34999: LET c5=PEEK
34993+256*PEEK 34994: GO TO 1
```

La opción 5, como las anteriores, debe utilizarse únicamente después de que se haya creado un fichero, por eso, la primera línea verifica que el contador de caracteres 'c' se ha desplazado del principio del fichero. A continuación tiene que leerse el nombre

del fichero en el buffer de línea. El usuario puede optar por cambiar el nombre del fichero antes de proceder a la grabación. La línea **5Ø3Ø** escribe el nuevo nombre en el lugar del anterior.

```
5ØØØ IF c=te THEN  GO TO 1
5ØØ5 CLS : PRINT " RECORDING FIL
E "'': LET z$="": FOR j=1 TO 1Ø:
 LET z$=z$+CHR$ PEEK (34879+j):
PRINT z$(j);: NEXT j
5Ø1Ø GO SUB 7Ø: IF b$="z" THEN
GO TO 1
5Ø2Ø IF b$<>"c" THEN  INPUT "Key
 new name ";z$: IF LEN z$ 1 OR L
EN z$>1Ø THEN  GO TO 5Ø2Ø
5Ø3Ø IF b$<>"c" THEN  FOR j=3488
Ø TO 34889: POKE j,32: NEXT j: F
OR j=1 TO LEN z$: POKE 34879+j,C
ODE z$(j): NEXT j: GO TO 5ØØØ
```

Se recuerda que la clavija de los auriculares debe retirarse antes de comenzar la grabación. La línea **5Ø5Ø** almacena el contador de caracteres y el número de registro ('c' y 'c5') antes de que se grabe el fichero. Si el usuario pulsara 'C' para verificar la grabación, las líneas **5Ø7Ø** y **5Ø8Ø** escribirían las instrucciones necesarias.

```
5Ø4Ø CLS : PRINT " PULL OUT THE
EAR PLUG "
5Ø5Ø RANDOMIZE c: POKE 34998,PEE
K q: POKE 34999,PEEK r: RANDOMIZ
E c5: POKE 34993,PEEK q: POKE 34
994,PEEK r
5Ø6Ø SAVE z$CODE li,c-33999: PRI
NT ''"Key C to verify ": INPUT b
$: IF b$<>"c" THEN  GO TO 1
5Ø7Ø PRINT '' FLASH 1;"REPLACE E
ARPLUG"
5Ø8Ø PRINT ''"Re-wind & start pl
aying": VERIFY z$CODE : GO TO 1
```

La opción 6 debe usarse antes que ninguna otra pero, sin embargo, como su uso no es muy frecuente, está situada al final del menú. Las rutinas de inserción, clasificación y búsqueda son las más frecuentemente usadas, por este motivo han sido ubicadas

cerca de la parte superior de la lista de opciones. La línea 6Ø2Ø llama a la rutina (2Ø) para verificar que el contenido de z$ es un número. El resto de la línea 6Ø2Ø rechaza las respuestas incorrectas dadas a la pregunta. A continuación se inicializan una serie de cadenas que almacenan los detalles del fichero y del formulario. Los elementos extras en la serie (a+1,1Ø) sirven para almacenar el nombre del fichero.

```
6ØØØ PRINT " FORM DESIGN "'''"How
 many fields?"
6Ø1Ø INPUT z$: IF z$="z" THEN  G
O TO 1
6Ø2Ø GO SUB 2Ø: IF x=1 OR z=Ø OR
 z>15 THEN  GO TO 6Ø1Ø
6Ø3Ø LET a=z: PRINT 'a;" fields"
': DIM c$(a+1,1Ø)
6Ø4Ø FOR j=1 TO a: PRINT "field
";j;TAB 1Ø; PAPER 6;"         ";
AT j+4,1Ø;
6Ø45 INPUT a$: IF a$="" THEN  GO
 TO 6Ø45
6Ø5Ø LET c$(j, TO 9)=a$: PRINT
PAPER 6;c$(j, TO 9);
6Ø55 INPUT "Length 1 or 2?";b$:
IF b$<>"1" AND b$<>"2" THEN  GO
TO 6Ø55
6Ø6Ø LET c$(j,1Ø)=b$: PRINT TAB
21;"length ";b$: NEXT j
6Ø7Ø INPUT "File Name?";c$(a+1):
 PRINT AT Ø,16; INVERSE 1;" ";c$
(a+1): GO SUB 7Ø: IF b$="c" THEN
  GO TO 62ØØ
```

Si los detalles del fichero fueran correctos, la siguiente sección se saltaría. Las siguientes líneas permiten eliminar los errores introducidos en el diseño. Como puede apreciarse, se ocupa más programa para corregir los posibles errores que para introducir los datos por primera vez. La variable 'x' toma el valor del número de la línea de la pantalla en la que el primer nombre de campo aparecerá impreso. Sobre esta línea se imprimirá el cursor intermitente. La rutina (1Ø) se usa para leer el teclado y la línea 61ØØ acepta únicamente dos teclas: 'z' y 'e'. 'z' mueve el cursor hacia abajo de la lista y 'e' sirve para editar el campo señalado por el cursor. Si la línea 613Ø empujase a 'x' más allá del final de la lista,

la línea 6140 le haría retroceder hasta la primera posición de la misma.

```
6080 LET x=5
6090 PRINT  FLASH 1;AT x,19;"▒";
6100 GO SUB 10: IF b$<>"z" AND b
$<>"e" THEN  GO TO 6100
6110 IF b$="e" THEN  GO TO 6160
6120 PRINT AT x,9;" ";
6130 LET x=x+1
6140 IF x=5+a THEN  GO TO 6080
6150 GO TO 6090
```

Las líneas 6160 y 6190 son repetición de las líneas de entrada de datos.

```
6160 LET j=x-4: INPUT a$: IF a$=
"" THEN  GO TO 6160
6170 LET c$(j, TO 9)=a$: PRINT A
T x,10; PAPER 6;c$(j, TO 9)
6180 INPUT "Length 1 or 2?";a$:
IF a$<>"1" AND a$<>"2" THEN  GO
TO 6180
6190 LET c$(j,10)=a$: PRINT AT x
,28;a$;AT x,9;" ": GO TO 6070
```

En este momento, la computadora contendrá suficiente información para diseñar un formulario para el nuevo fichero. Esta labor requiere algún tiempo, ya que es un trabajo bastante extenso en BASIC, pero no es preciso introducirlo en lenguaje máquina, dado que su utilización es poco frecuente. Un mensaje intermitente se imprime durante el compás de espera para tranquilizar e informar al usuario que algo está ocurriendo y que la tardanza no se debe al fallo de la computadora. El primer carácter de la cadena de la línea 2 ocupa en la memoria la posición 23 779. La línea 6210 da a 'y' el valor de esta dirección y el resto de la rutina mueve 'y' a otro lugar de la cadena en base a los parámetros del fichero. Los primeros nueve caracteres se ocupan por el nombre del primer campo, seguido de suficientes espacios para la longitud de dicho campo. La variable 'x' cuenta el número de campos largos del formulario y la línea 6225 envía a la computadora a las rutinas de edición en el caso de que haya demasiados para que quepan en la pantalla.

```
6200 PRINT ' FLASH 1;"Logging Fi
le Details"
6210 LET y=34200: LET x=0: FOR j
=1 TO a: RANDOMIZE y-34200+23779
: POKE 34849+j*2-1,PEEK q: POKE
34849+j*2,PEEK r: POKE 34889+j,V
AL c$(j,10)
6215 LET y1=y-34200: LET y1=y1-3
2*INT (y1/32)
6220 FOR k=1 TO 9: POKE y,CODE c
$(j,k): LET y=y+1: NEXT k: IF c$
(j,10)="2" THEN  LET x=x+1:
6225 IF x+a>20 THEN  PRINT AT 20
,6;"Too many ""2"" fields": GO T
O 6080
6230 LET z=7: LET y1=y-34200: LE
T z=55: IF c$(j,10)="1" THEN  LE
T z=23
6250 FOR k=1 TO z: POKE y,32: LE
T y=y+1: NEXT k
```

Dentro de esta sección, todavía hay que almacenar la longitud del formulario en las posiciones 34 995 y 34 996 y el nombre del fichero a partir de la posición 34 880. El número de registro comienza valiendo 1 y el contador de caracteres toma el valor correspondiente al principio del buffer, después el control se devuelve al menú. La línea 7000 no precisa explicación.

```
6260 NEXT j: RANDOMIZE y-34200:
POKE 34995,PEEK q: POKE 34996,PE
EK r: POKE 34997,a
6270 FOR j=1 TO 10: POKE 34879+j
,CODE c$(a+1,j): NEXT j
6280 LET c=te: LET c5=1: GO TO 1
7000 PRINT AT 9,9;"END OF PROGRA
M"
```

Al final del capítulo hay una sección destinada a los usuarios que planeen modificaciones al programa. Los detalles completos sobre la información contenida en el buffer de línea están allí, junto con el contenido de cada variable y el propósito de cada subrutina utilizadas en el programa. Esto nos lleva a la última parte del programa, el conjunto de subrutinas comprendidas entre la línea 2 y la línea 100. Se dedica un capítulo a las rutinas escritas en

código máquina. La primera rutina es una línea, ¡pero qué línea! La cadena de la línea 2 consta de 640 espacios, suficiente para el más largo de los formularios que producirá este programa. Si se dispusiera de una copia de una línea idéntica utilizada en SPEC-TEXT, se ahorraría tiempo. En caso contrario, si el procesador de textos fuera a ser grabado más tarde, sería conveniente grabar las líneas 1 y 2 ya que son idénticas en ambos programas y precisan mucho tiempo para introducirse en la computadora. Comenzar introduciendo la línea:

2 LET a$=“   ”: RETURN

y mover el cursor hacia atrás hasta situarle entre las dos dobles comillas, mantener pulsada la tecla del espaciador durante un largo rato hasta que se considere que se tienen suficientes espacios. A continuación teclear ENTER y después GO SUB 2 seguido de PRINT LEN a$. Si la computadora no respondiera exactamente 640, se debería continuar ajustando la línea hasta que se obtuviera la respuesta correcta.

```
1 GO TO 900
2 LET a$="




                    ": RETURN
```

La línea 3 no es una subrutina sino que es la línea en la que la computadora encuentra información vital cuando se carga el programa desde cinta. Para grabar la versión final del programa deberá usarse la siguiente instrucción:

SAVE “fichero” LINE 3   (véase el manual para más detalles)

de este modo, todas las variables necesarias se inicializarán automáticamente cada vez que se cargue el programa.

```
3 LET te=35ØØØ: LET li=34ØØØ:
 LET q=2367Ø: LET r=23671: LET c
=te: LET c2=596ØØ: GO TO 1
```

Las líneas comprendidas entre la 1Ø y la 12 leen el teclado, trasfieren la pulsación a b$ y producen el sonido característico del teclado.

```
  1Ø IF INKEY$<>"" THEN  GO TO 1
Ø
  11 LET b$=INKEY$: IF b$="" THE
N  GO TO 11
  12 BEEP .ØØ3,Ø: RETURN
```

El sistema más seguro para introducir información numérica en un programa sin que se puedan producir fallos, es utilizar primero una cadena para almacenar dicha información y después verificar que el contenido de dicha cadena es estrictamente numérico. La rutina siguiente verifica los caracteres de z$ colocando la bandera 'x' en 1 si se encontrasen caracteres no numéricos. Si después del bucle de verificación, 'x' todavía vale cero la cadena podrá ser traducida a un número real, sin problemas, después se almacenará el número en la variable 'z'. Esta rutina sólo sirve para números enteros. Los números decimales necesitarían otra verificación.

```
  2Ø LET x=Ø: IF LEN z$=Ø THEN
LET x=1: RETURN
  21 FOR j=1 TO LEN z$: IF z$(j)
<"Ø" OR z$(j)>"9" THEN  LET x=1
  22 NEXT j: IF x=1 THEN  RETURN

  23 LET z=VAL z$: RETURN
```

Las dos rutinas siguientes no son tan complicadas como parece a primera vista. En primer lugar, se utiliza la rutina de la línea 85 para asegurarse que el contador de caracteres apunta hacia el interior del buffer. Si se aplica RANDOMIZE a un número, la computadora lo convierte en un formato de dos bytes y almacena el resultado en las posiciones 23 67Ø y 23 671. El número se divide por 256, almacenándose el cociente en 23 67Ø y el resto en 23 671. Las variables 'p' y 'q' contienen respectivamente los valores 23 67Ø y 23 671, de forma que la función de la línea 3Ø es convertir los valores de 'v1', 'v2' y 'v3' en un formato de dos bytes

almacenando el resultado en la zona destinada al código máquina. Así se prepara el código máquina para trasladar 'v1' bytes a la posición 'v2' desde la posición 'v3'.

```
  3Ø GO SUB 85: RANDOMIZE v1: PO
KE 6ØØØ1,PEEK q: POKE 6ØØØ2,PEEK
 r: RANDOMIZE v2: POKE 6ØØØ4,PEE
K q: POKE 6ØØØ5,PEEK r: RANDOMIZ
E v3: POKE 6ØØØ9,PEEK q: POKE 6Ø
Ø1Ø,PEEK r: RANDOMIZE USR 6ØØØØ:
 RETURN
  35 GO SUB 85: RANDOMIZE v1: PO
KE 6ØØ15,PEEK q: POKE 6ØØ16,PEEK
 r: RANDOMIZE v2: POKE 6ØØ18,PEE
K q: POKE 6ØØ19,PEEK r: RANDOMIZ
E v3: POKE 6ØØ23,PEEK q: POKE 6Ø
Ø24,PEEK r: RANDOMIZE USR 6ØØ14:
 RETURN
```

La línea 35 realiza un trabajo similar por medio de una rutina preparada para efectuar traslaciones de bloques de memoria hacia posiciones superiores. La rutina (4Ø) imprime un formulario blanco en la pantalla. Primero se imprime el nombre del fichero (almacenado a partir de la posición 34 88Ø) en la cabecera de formulario. El número de registro y el número de bytes que quedan en el buffer se imprimen en la parte inferior. La línea 42 lee las posiciones en la pantalla en el buffer de línea y la línea 43 imprime bloques en negro en el lugar de las cabeceras.

```
  4Ø LET a=PEEK 34997: LET b=348
8Ø: PRINT AT Ø,1Ø; INVERSE 1;" "
;: FOR L=b TO b+9: PRINT  INVERS
E 1;CHR$ PEEK L;: NEXT L: PRINT
 INVERSE 1;" "
  41 PRINT AT 21,Ø;"record ";c5;
"  ";TAB 13;"file space ";c2-c;"
 "
  42 FOR L=1 TO a: LET z=PEEK (3
4848+L*2)+256*PEEK (34849+L*2)-2
3715: LET m=INT (z/32): LET n=z-
m*32
  43 PRINT  OVER 1;AT m-1,n; INV
ERSE 1;"          ": NEXT L
  44 RETURN
```

La rutina (45) localiza los nombres de los campos en el buffer de línea y los ubica en la cadena de la línea 2 por medio de la rutina (3Ø). Debe tenerse en cuenta que algunas de las rutinas están interrelacionadas.

```
  45 LET v1=PEEK 34995+256*PEEK
34996: LET v2=23779: LET v3=342Ø
Ø: GO SUB 3Ø: RETURN
```

Las rutinas (5Ø) y (55) imprimen la cadena de la línea 2 en la pantalla. La rutina (55) imprime 'OVER 1' sin interferir con lo que está escrito en la pantalla.

```
  5Ø GO SUB 2: PRINT AT 1,Ø;a$:
RETURN
  55 GO SUB 2: PRINT  OVER 1;AT
1,Ø;a$: RETURN
```

La rutina (6Ø) imprime los registros en la pantalla de acuerdo con el formato diseñado mediante la opción 6. Dicha rutina imprime un registro tras otro hasta que se alcanza el final del fichero o el usuario indique a la computadora que se detenga. En esta sección existen grandes posibilidades de modificaciones del programa. La secuencia de llamadas a subrutinas que aparece aquí es tan solo una de las muchas posibilidades, otras combinaciones podrían adaptarse mejor a necesidades particulares. En el capítulo dedicado al uso de estos programas, se dan algunas sugerencias para la modificación de la rutina (6Ø). La línea 6Ø utiliza la rutina DOKE ubicada en la línea 95 llamando después a la rutina que imprime el formulario en blanco. La línea 61 limpia la cadena de la línea 2, almacena en ella las etiquetas de los campos y hace entrar en acción el código máquina para introducir un registro. El registro se imprime en la pantalla por medio de la rutina (5Ø). La rutina (42) suprime las zonas negras de la pantalla con objeto de llamar la atención sobre el cambio experimentado en la misma. Esto es necesario porque dicho cambio podría no ser evidente si los registros consecutivos fueran similares.

```
  6Ø RANDOMIZE te: LET d=6Ø496:
GO SUB 95: GO SUB 4Ø
  61 GO SUB 1ØØ: GO SUB 45: RAND
OMIZE USR 6Ø5ØØ: GO SUB 5Ø: GO S
UB 42
```

La línea 62 utiliza la rutina DOKE (95) para llevar el puntero del código máquina al comienzo del siguiente registro, después examina si se ha alcanzado el final del fichero. Las líneas 63 y 64 permiten al usuario realizar una copia de la pantalla o volver al menú principal. La línea 65 envía a la computadora en busca de otro registro.

```
  62 INPUT z$: LET d=6Ø496: GO S
UB 9Ø: LET c3=d+1: RANDOMIZE c3:
 LET d=6Ø496: GO SUB 95: IF c3>=
c THEN RETURN
  63 IF z$="c" THEN  COPY
  64 IF z$="z" THEN  RETURN
  65 GO TO 61
```

Si la rutina (7Ø) es la más simple de todas, la rutina (8Ø) es más complicada de lo que parece a primera vista. Esta rutina utiliza la rutina DOKE para inicializar otra rutina escrita en lenguaje máquina que sirve para encontrar el principio y el final del registro señalado por las posiciones de memoria 6Ø 135 y 6Ø 136. Antes de volver al programa principal, se utiliza la rutina (61) para imprimir el registro en la pantalla. No se utiliza la rutina (6Ø) porque la primera línea coloca el puntero al principio del buffer (dirección 35 ØØØ).

```
  7Ø INPUT "Key C if OK ";b$: RE
TURN
  8Ø RANDOMIZE c: LET d=6Ø567: G
O SUB 95: RANDOMIZE USR 6Ø57Ø: P
OKE 6Ø496,PEEK 6Ø135: POKE 6Ø497
,PEEK 6Ø136: GO SUB 61: RETURN
```

La rutina (85) se explica por sí sola y las rutinas (9Ø) y (95) son las rutinas DEEK y DOKE. La rutina (9Ø) devuelve en 'd' el valor contenido en las dos posiciones hacia las que apuntaba 'd' cuando la rutina fue llamada. La rutina (95) almacena en las posiciones hacia las que apunta 'd' los valores almacenados en las variables 'q' y 'r' por la instrucción RANDOMIZE.

```
  85 IF c<te THEN  LET c=te
  86 RETURN
  9Ø LET d=PEEK d+256*PEEK (d+1)
: RETURN
```

```
  95 POKE d,PEEK q: POKE d+1,PEE
K r: RETURN
```

La rutina (1ØØ) prepara el código máquina para rellenar con espacios la cadena de la línea 2.

```
 1ØØ RANDOMIZE 23779: POKE 6Ø152
,PEEK q: POKE 6Ø153,PEEK r: RAND
OMIZE 24418: POKE 6Ø155,PEEK q:
POKE 6Ø156,PEEK r: POKE 6Ø158,32
: RANDOMIZE USR 6Ø151: RETURN
```

El programa BASIC está completo y debe grabarse en cinta de modo que se autoejecute a partir de la línea 3. *El programa requiere que el código máquina esté situado a partir de la posición 6ØØØ.* Si la computadora comienza a trabajar con el programa y el código máquina no estuviera en su lugar, es muy posible que se perdiera el control, en cuyo caso habría que apagar la computadora y volver a cargar el programa en BASIC y el código máquina. En el próximo capítulo se explica cómo utilizar el programa, mientras tanto, a continuación, se dan unos detalles de mucha utilidad para cualquier modificación o desarrollo del programa.

Composición de buffer de SPECFILE

34 ØØØ a 34 199	Texto de reserva para el examen y edición de la información
34 2ØØ a 34 839	Texto del formulario en blanco
34 85Ø a 34 879	Posición de cada campo en la pantalla
34 88Ø a 34 889	Nombre del fichero
34 89Ø a 34 9Ø4	Longitud de los campos
34 993 y 34 994	Número de registros en el fichero (c5)
34 995 y 34 996	Longitud del formulario
34 997	Número de campos por registro
34 998 y 34 999	Longitud del fichero (c1)

8 INSTRUCCIONES OPERATIVAS DE SPECFILE

Las instrucciones operativas a las que se dedica este capítulo sirven para familiarizar a los usuarios de SPECFILE con su manejo y también como recordatorio para aquellos otros que no lo hayan utilizado durante algún tiempo. Las instrucciones son lo suficientemente completas y claras como para ser comprendidas por una persona apenas iniciado en el mundo de las computadoras.

Al cargar el programa lo primero que aparece en la pantalla es un menú de opciones, pero sólo tres de ellas pueden utilizarse en principio. Si se pulsa 1, 2, 3 y 5 se producirán unas perturbaciones momentáneas en la pantalla. La razón de esto es que no tiene ningún sentido tratar de clasificar o de encontrar registros en un fichero que todavía no existe. Estas opciones podrán utilizarse más adelante. La primera de estas opciones (1) corresponde a la introducción de datos en un fichero y puede utilizarse tan pronto como se haya creado el fichero. Las opciones 4 y 6 sirven para la creación de un fichero. Como lo más probable es que en principio no dispongamos de ningún fichero tendremos que usar en primer lugar la opción 6 con la que podemos definir la estructura de un fichero y el formulario para la aparición de los registros en pantalla.

PARADA

Pulsando la opción 7 aparecerá en pantalla el mensaje 'EN OF PROGRAM' cuyo significado es evidente. El motivo de incluir una opción como ésta es que de este modo se facilita la carga de otros programas. Para ejecutar el programa de nuevo, basta teclear:

GO TO 1 y después pulsar ENTER

En caso de pulsar RUN por error, podría perderse parte de la información almacenada. No obstante, se podría recuperar con:

GO TO 3 y después pulsar ENTER

De esta forma volveremos a la misma situación que cuando los datos fueron cargados desde la cinta. No es fácil que el progra-

ma deje de funcionar accidentalmente, a pesar de lo cual podría suceder. De todas formas, con las dos instrucciones citadas anteriormente, el lector está en disposición de resolver la mayor parte de los problemas de este tipo que pueden presentarse.

DISEÑO DE UN NUEVO FICHERO

Al pulsar la opción 6, tenemos que informar a la computadora del número de partes de que consta el registro. Estas partes reciben el nombre de campos y cada registro puede tener hasta quince. Cada campo dispondrá de una cabecera en la pantalla para la entrada de datos. Para simplificar crearemos un fichero de 4 campos. La mayor partes de las veces tendremos que confirmar a la computadora nuestras decisiones pulsando ENTER, como en este caso. A continuación la computadora responderá imprimiendo un recuadro gris (amarillo en los televisores de color) destinado a la cabecera de campo. Después, en la parte inferior izquierda de la pantalla la computadora nos pedirá que escribamos el nombre del campo. Tan solo los nueve primeros caracteres pasarán a formar parte del nombre. Inmediatamente después de pulsar ENTER para confirmar el nombre del campo, el mismo se escribe en el rectángulo gris. A continuación, es preciso informar a la computadora si el formato del campo será corto (hasta 23 caracteres) o largo (hasta 55 caracteres); esta elección se realiza pulsando 1 ó 2 y después ENTER y aparecerá impresa en la pantalla.

El proceso de introducir el nombre y el formato debe repetirse para los cuatro campos que componen nuestro fichero. Si se cometieran errores durante el proceso de diseño, no es preciso volver a comenzar, puesto que hay una oportunidad de efectuar correcciones antes de finalizar. Inmediatamente después de introducir el formato del último campo, la computadora preguntará el nombre del fichero e imprimirá el formulario correspondiente en la pantalla. Después pedirá al usuario que pulse ‘C’ si está de acuerdo, en cuyo caso, volverá al menú. Pero supongamos que hemos cometido un error de escritura en el cuarto campo, en ese caso pulsando ENTER sin haber pulsado ‘C’ aparecerá un cursor negro en la pantalla junto al primer nombre de campo. Este cursor puede desplazarse verticalmente mediante ‘Z’. Debemos colocar el cursor sobre el cuarto campo y pulsar ‘E’ para editarlo. En este momento, en la parte inferior de la pantalla aparecerá el nombre del campo y podremos corregirlo. Tendremos que escribir de nuevo el nombre del campo y el formato, después la computadora nos pedirá que pulsemos de nuevo la tecla ‘C’ si todo es correcto. Al pulsar ‘C’ se hará una pequeña pausa duran-

te la cual la computadora almacenará el nuevo diseño y entretanto un mensaje intermitente en la pantalla indicará que el proceso está en marcha. A continuación la computadora retornará al menú. Es posible modificar la estructura y formulario de un fichero en cualquier momento, pero la información se perderá. Concluido el diseño del fichero podremos utilizar la opción 1, pero todavía no es posible utilizar las opciones 2, 3 y 5, porque el fichero necesita tener al menos un registro antes de que pueda clasificarse, escudriñarse o grabarse en cinta.

ENTRADA DE DATOS

Pulsando desde el menú la tecla '1' la computadora imprimirá un formulario en blanco para cumplimentarse. En la parte inferior de la pantalla aparecerá el número del registro y el espacio disponible. Al principio la capacidad será de 24 600 caracteres y cuando esté llena la computadora emitirá una señal auditiva y rechazará nuevas entradas. El cursor intermitente indica el lugar de la pantalla donde aparecerán los caracteres. Comience a escribir y se dará cuenta de que no es posible introducir más de 23 caracteres en un campo corto ni más de 55 en un campo largo. Mediante las teclas SHIFT y Ø pueden borrarse hacia atrás los caracteres de un campo. Cuando consideremos correcto el contenido del primer campo para pasar al siguiente campo pulsaremos ENTER, se producirá un ligero sonido y el cursor reaparecerá de nuevo en el siguiente campo. Este proceso debe repetirse hasta completar el registro. Al pulsar ENTER por última vez escucharemos un doble sonido y dispondremos de tres opciones.

Pulsar 'C' para confirmar el registro y éste se almacenará en la memoria, a continuación aparecerá un nuevo formulario en la pantalla. Si se pulsa tan solo la tecla ENTER, la computadora asumirá que hay un error y rechazará el registro imprimiendo un nuevo formulario en blanco. Pulsando 'Z' y después ENTER la computadora rechazará el registro y volverá al menú. De este modo, puede abandonarse la opción 1 (entrada de datos). El sistema de sonidos elegido es el siguiente: un sonido agudo significa que el dato ha sido aceptado, por el contrario un sonido grave significa que el dato ha sido rehusado. Estas señales auditivas son de mucha utilidad durante la entrada de datos.

CLASIFICACION

Para seleccionar esta opción, pulse la tecla '2' desde el menú. A continuación la computadora preguntará por qué campo debe-

rá realizar la clasificación. El programa permite la clasificación del fichero por cualquier campo excepto por el último. En el Capítulo 11 se sugieren algunos usos de esta facilidad.

Al introducir el número del campo el primer dato ordenado de la lista aparecerá inmediatamente. Con el fichero completo la computadora precisará de algunos segundos para efectuar la clasificación, sin embargo, al principio con el fichero casi vacío, el proceso será inmediato. Manteniendo pulsada la tecla ENTER los registros irán apareciendo en el orden solicitado. En cualquier momento puede abandonarse el proceso pulsando la tecla 'Z' antes de ENTER. Tanto si se pulsa 'Z' como si se alcanza el final del fichero, la computadora solicitará un nuevo campo para efectuar una nueva clasificación. Para abandonar esta opción pulsar ENTER. Mientras la computadora está imprimiendo los registros en orden en la pantalla, se puede obtener una copia en papel pulsando la tecla 'C'.

BUSQUEDA Y EDICION

Mediante la tecla '3' se selecciona la opción de búsqueda y tendremos que decir a la computadora lo que debe localizar. Escriba aquello que se desea buscar y si existiera una ocurrencia de lo buscado en el fichero, el registro correspondiente aparecería en la pantalla. En el caso de que el fichero no contenga el grupo de caracteres que el usuario ha solicitado la computadora informará que en su búsqueda ha llegado al final del fichero. En este caso pulsando la tecla ENTER volverá de nuevo al menú. Si la información se hubiera encontrado y como consecuencia tuviéramos un registro en pantalla, mediante la tecla ENTER nos desplazaríamos al siguiente registro.

Si se pulsa la tecla 'Z' antes que ENTER, la computadora nos ofrecerá tres alternativas. La primera DELETE (borrar), se imprime al pie de la pantalla. Pulsar ENTER y la segunda alternativa COPY (copiar) se imprimirá sobre la anterior. Pulsando ENTER de nuevo se imprime la tercera opción ESCAPE (salida). Es posible pasar de una opción a otra cíclicamente hasta que la que deseemos aparezca en la pantalla. Para elegir una opción, bastará pulsar cualquier otra tecla antes de ENTER. Con COPY se obtiene una copia en papel de la pantalla, mediante ESCAPE se vuelve al menú y con DELETE se borra el registro del fichero. Se ha previsto un sistema para evitar el borrado accidental de un registro. Cuando se intente borrar un registro la computadora preguntará 'ARE YOU SURE?' (¿está seguro?) y será preciso confirmar nuestra decisión escribiendo 'yes' para que el registro se borre efectivamente.

GRABACION DE UN FICHERO

Al pulsar '5' en el menú, la computadora se prepara para efectuar la grabación del fichero, para ello en primer lugar imprimirá en la pantalla el nombre del fichero tal y como se le bautizó en principio. Si se quiere cambiar el nombre antes de la grabación, pulse ENTER sin haber pulsado previamente 'C' y la computadora solicitará el nuevo. Sólo se aceptan nombres con un número máximo de 10 caracteres. Cuando nos dispongamos a efectuar la grabación la computadora nos recordará que debemos cambiar las clavijas antes de comenzar. Puede verificarse la grabación con el contenido de la memoria antes de que la computadora vuelva al menú. Para verificar pulse 'C' y siga las instrucciones de la pantalla.

CARGA DE UN FICHERO GRABADO EN CINTA

La computadora precisa que se escriba el nombre exacto del fichero que se pretende cargar. Si nos equivocamos en un solo carácter del nombre y pulsamos 'C', el único modo de escapar es pulsar las teclas SHIFT y BREAK al mismo tiempo. De este modo el programa se detendrá y para volver a ejecutarlo será necesario escribir:

GO TO 1 y después ENTER

Si el nombre fuera correcto la computadora cargará el primer fichero que localice con ese nombre. El grabado en cinta está tan perfeccionado que es difícil que nos encontremos con algún problema, no obstante si tuviéramos algún error de carga ejecutando 'GO TO 1' solucionaríamos el problema. Inmediatamente después de cargado el fichero la computadora volverá al menú.

MENSAJES DE ERROR

La rutina en lenguaje máquina que se ocupa de clasificar el fichero puede producir algunos mensajes de error. Si se hubiera deteriorado un fichero o tuviera un número insuficiente de registros, se produciría un mensaje de error al intentar clasificarlo. La perfección del sistema de grabación a la que hacíamos mención anteriormente, hace difícil que pueda deteriorarse un fichero, sin embargo, hay otras fuentes de errores como por ejemplo, intentar clasificar por un campo que en algunos registros pueda estar vacío. Debería dejarse el último campo para aquellos atributos o

características que tengan lugar ocasionalmente. Por este motivo no se puede clasificar el fichero por el último campo.

CONSEJOS OPERATIVOS

A pesar de que como hemos comentado los errores son raros, pueden producirse tanto en la carga desde cinta como por fallos en el suministro de corriente. Por ello es recomendable realizar frecuentes copias de seguridad de los ficheros, de esta forma, si apareciera algún problema no se perdería la totalidad de la información. Utilizar un cassette para cada fichero. Cuando la memoria de la computadora está llena, se necesitan alrededor de dos minutos para grabar la información, por lo tanto es conveniente utilizar cintas de unos tres minutos de duración por cada cara. Las cintas C90 necesitan demasiado tiempo para rebobinarse. Cada fichero debería grabarse en dos cintas y debe actualizarse la grabación cada vez que se usa el fichero.

Planificar bien la estructura de cada fichero dejando en blanco uno o dos campos para almacenar la posible información que necesitemos en el futuro. En todo momento se puede modificar la estructura del fichero pero se perdería toda la información introducida hasta ese momento. Para conservar la estructura y borrar todos los registros puede procederse a borrar los registros uno por uno, o bien, se podría elegir la opción 7 del menú y escribir:

GO TO 3 y después ENTER

En caso de que en alguna ocasión se confunda en la selección de la opción del menú, generalmente se puede escapar pulsando 'Z'. En las explicaciones de las opciones individuales se detalla cómo volver al menú en cada caso concreto. Pronto alcanzará un dominio considerable del programa y de la manipulación de las teclas 'Z' y ENTER para su control.

En el Capítulo 11 se sugieren diversas utilizaciones de los programas de este libro, no obstante, conviene dar aquí algunos consejos sobre el particular. La rutina de clasificación ordena los elementos de un fichero de acuerdo con los códigos internos de los caracteres. Los números se clasifican de acuerdo con el número de dígitos que posean, por ejemplo Ø97 precederá a 236 pero, sin embargo, 97 sin el cero delante, se colocaría después. La computadora da preferencia a los caracteres numéricos sobre los alfabéticos y a las letras mayúsculas sobre las minúsculas. Si se están utilizando simultáneamente mayúsculas y minúsculas, la búsqueda de palabras debe hacerse prescindiendo de su primera letra. La computadora encontrará 'áquina' con la misma facilidad

con que encontraría 'Máquina' y además, no será necesario buscar 'máquina' a continuación. El programa es más rápido si primero se clasifica el fichero y después se realizan las búsquedas.

VARIABLES Y SUBRUTINAS DE SPECFILE

El programa utiliza un pequeño número de variables, la mayor parte de las cuales se utilizan para más de una función. De este modo se pretende disponer del mayor espacio posible para el fichero propiamente dicho. La lista que sigue a continuación contiene las variables más importantes. No se incluyen las que se utilizan menos frecuentemente y para realizar trabajos menos importantes.

Variables

a$	—contiene la parte del texto que aparece en la pantalla. Esta genera por la subrutina de la línea 2 y se utiliza en ocasiones como almacen temporal
b$ y z$	—almacenamientos temporales
c$	—cadena utilizada para contener los nombres y formatos de los campos del fichero durante el proceso de creación
a	—número de campos de cada registro
b	—cuenta los caracteres de la pantalla durante la entrada de datos
b1	—límite del valor de b
c	—contador de caracteres del buffer
c2	—límite del buffer
c3 y c4	—contadores temporales de caracteres
c5	—número de registros en el fichero
d	—variable utilizada por las rutinas DEEK y DOKE
f	—bandera de propósito general
j, k y l	—variables de control en bucles
m y n	—coordenadas en la pantalla
q y r	—utilizadas por las rutinas DEEK y DOKE
te	—35 000, comienzo del buffer
li	—34 000, comienzo del buffer de línea
v1, v2 y v3	—variables utilizadas para dirigir el código máquina
x	—bandera utilizada por la rutina de verificación

Subrutinas

Rutina 2—inicializa la variable a$. Esta cadena se carga con parte del contenido del buffer mediante una rutina escrita en lenguaje máquina.

Rutina 3—aunque no es propiamente una subrutina, es la primera línea que se ejecuta al cargar el programa. Inicializa las variables te, li, q, r, c y c2.

Rutina 10—lee el teclado y carga el resultado de la primera pulsación en b$.

Rutina 20—verifica z$ y coloca una bandera x a 1 si el contenido de esta variable no es un número, en caso contrario la bandera x toma el valor 0.

Rutina 30—es una rutina conductora del código máquina que lleva hacia abajo v1 bytes desde la dirección v3 a la dirección v2.

Rutina 35—similar a la anterior pero para trasladar los bloques hacia posiciones superiores en la memoria.

Rutina 40—imprime un formulario en blanco en la pantalla.

Rutina 45—imprime las etiquetas en el formulario preparado por la rutina anterior.

Rutina 50—imprime el contenido de la cadena de la línea 2.

Rutina 55—efectúa la misma función que la rutina anterior pero esta imprime 'OVER 1'.

Rutina 60—imprime en la pantalla los registros ordenados hasta llegar al final del fichero, o bien, hasta que el usuario le indique lo contrario.

Rutina 70—escribe 'Key C if OF' y lleva las pulsaciones a b$.

Rutina 80—localiza el principio y el final de los registros encontrados por la rutina FIND. Esta rutina llama a otra escrita en lenguaje máquina que almacena en las posiciones 60 135 y 60 136 la dirección del principio del registro y en las posiciones 60 565 y 60 566 la dirección del final. Si FIND alcanzase los límites del almacenamiento la bandera 60 569 se colocaría a '1'.

Rutina 90—es una copia DEEK. La dirección contenida en la variable 'd' se trasforma en un formato de dos bytes y se trasfiere de nuevo a 'd'.

Rutina 95—es una copia de DOKE. Las dos posiciones hacia las

que apunta 'd' se cargan con los resultados de PEEK q y PEEK r.

Rutina 100—esta última rutina completa la cadena de la línea 2 con espacios.

9 CODIGO MAQUINA DE SPECFILE

SPECFILE tiene necesidades extras que no pueden ser satisfechas con la rapidez necesaria por el BASIC. El código máquina creado para SPECTEXT puede utilizarse para SPECFILE pero es preciso ampliarlo para cumplir con todos los requerimientos. Aquí se dispone de tres nuevas rutinas START, DUMP y SORT. Las dos primeras son sencillas, mientras que la tercera es más compleja y poderosa. START sirve para localizar el comienzo y el final de un registro. La rutina FIND encontrará un grupo de caracteres en el fichero pero no proporciona las direcciones del principio y final del registro correspondiente. DUMP toma el registro dado por START y lo coloca en un formulario. De esta forma, los datos se presentan siempre en el formato diseñado por el usuario. La tercera rutina (SORT) proporciona al programa la potencia y velocidad necesarias en una clasificación. Esta rutina clasificará el fichero por cualquier campo. El algoritmo de clasificación utilizado es una simple modificación del 'bubble sort' (clasificación por burbujas).

Las tres rutinas se han desarrollado con el procedimiento del Capítulo 6. Los listados en BASIC, a partir de los cuales se desarrollaron las rutinas, se presentan junto con la documentación. Después viene la traducción a los códigos de operación del microprocesador Z80. Al final del capítulo se da el código numérico para aquellos usuarios que no dispongan de un ensamblador. Los programas en BASIC no es preciso introducirlos, se ofrecen tan solo para que el usuario pueda comprender mejor el proceso. Leyendo conjuntamente los dos bloques de código se observará lo similares que son los códigos de operación a sus equivalentes en BASIC. Para evitar gran cantidad de líneas que almacenen o lean números en un formato de dos bytes se han simulado las posiciones de memoria con variables como 'm1' 'm2', etc.

START es la rutina más sencilla y por tanto la más apropiada para empezar. Antes de que se pueda llamar a esta rutina, 'm2' debe contener una dirección y 'm3' debe apuntar hacia el final del fichero. Si la dirección corresponde a un lugar dentro del fichero, después de llamar a la rutina, la bandera 'm1' se devolverá con cero y 'm2' y 'm4' contendrán las direcciones del principio y final del registro. Si la dirección estuviera fuera de los límites del fiche-

ro, la bandera tomaría el valor 1. La mayoría de los registros (todos excepto el primero) contienen el marcador CHR$ 139 al principio y al final. Por tanto lo que hay que hacer es retroceder desde la dirección contenida en 'm2' hasta encontrar un marcador como éste y después avanzar hasta localizar el siguiente. Puede haber dos problemas. Primero puede que la dirección no corresponda a una posición en el interior del fichero y en segundo término, si el registro fuese el primero no sería posible localizar el marcador del principio.

Las primeras líneas del programa en BASIC inicializan las direcciones que la computadora utilizará como etiquetas. Este es el sistema de trabajo de muchos ensambladores. El trabajo principal comienza con la rutina de la línea 2Ø que carga cero en pl acumulador para que la línea 3Ø inicialice la bandera 'm1' con cero.

```
 1 REM BASIC version of START,
       for demonstration only,
       not to be entered
1Ø LET LA=5Ø
11 LET LB=17Ø
12 LET LC=2ØØ
13 LET LD=19Ø
14 LET LE=32Ø
2Ø LET a=Ø
3Ø LET m1=a
```

A continuación, el par de registros hl toma el valor de la dirección desde la cual debe comenzar la búsqueda. El par bc toma el valor de la dirección del principio del buffer. Estas dos posiciones de memoria se comparan cada vez que se empieza un ciclo del bucle. Si coincidieran antes de que se encontrase el CHR$ 139, el registro sería el primero del fichero. Si el resultado de la comparación fuera negativo, significaría que la dirección corresponde a una posición anterior al comienzo del fichero. Estas condiciones necesitan un tratamiento especial y las líneas 8Ø y 9Ø se encargan de enviar el control hacia las partes apropiadas del programa. La línea 6Ø contiene una sentencia REM puesto que es difícil simular en BASIC a la instrucción 'and a' del código máquina que permite limpiar todas las banderas antes de usar las instrucciones 'adc' y 'sbc'.

```
4Ø LET hl=m2
5Ø LET bc=35ØØØ
6Ø REM clear the results of
       previous calculations
```

```
70 LET hl=hl-bc
80 IF hl=0 THEN  GO TO LD
90 IF hl<0 THEN  GO TO LE
```

Ahora, la computadora tiene que retroceder en el fichero comparando cada carácter con el CHR$ 139 (marcador de fin de registro). El par de registros hl toma el valor de la dirección contenida en 'm2' y el acumulador se carga con el valor del carácter almacenado en la posición hacia la que apunta hl. Si no se encontrase el carácter CHR$ 139 la dirección contenida en hl se disminuye en una unidad y se almacena en 'm2' de nuevo. La línea 160 comienza un nuevo ciclo del bucle.

```
100 LET hl=m2
110 LET a=PEEK hl
120 LET a=a-139
130 IF a=0 THEN  GO TO LB
140 LET hl=hl-1
150 LET m2=hl
160 GO TO LA
```

El comienzo del registro es el carácter siguiente al CHR$ 139 que marca el final del anterior, por ello, las líneas 170 y 180 sitúan esta dirección en 'm2'. La línea 190 puede parecer superflua pero es necesaria para el funcionamiento del bucle.

```
170 LET hl=hl+1
180 LET m2=hl
190 LET hl=m2
```

La computadora se desplazará ahora hacia adelante en busca de un nuevo marcador que señalará el final del registro. La dirección de este marcador tiene que almacenarse en 'm3' pero también tiene que compararse en todo momento con la dirección del final del fichero. Si 'm3' llegase a apuntar más allá del final del fichero, la dirección contenida en principio en 'm2' correspondería a una posición situada fuera del buffer y la bandera 'm1' tendría que tomar el valor 1. Esto sucede cuando la línea 250 da un resultado mayor o igual a cero.

```
200 LET hl=hl+1
210 LET m3=hl
220 LET bc=m4
230 REM clear the results of
        previous calculations
```

```
240 LET hl=hl-bc
250 IF hl>=0 THEN  GO TO LE
```

Las siguientes líneas cargan, a lo largo del texto, cada carácter en el acumulador. Cuando el acumulador contiene el carácter 139, el trabajo ha terminado y 'm3' tiene la dirección del final del registro.

```
260 LET hl=m3
270 LET a=PEEK hl
280 LET a=a-139
290 IF a=0 THEN  RETURN
300 LET hl=m3
310 GO TO LC
```

Si la dirección contenida en principio por 'm2' estuviera fuera de los límites del buffer, el camino seguido por el programa sería distinto. Las líneas 320 y 340 forman parte de este camino alternativo y colocan la bandera 'm1' a 1.

```
320 LET a=1
330 LET m1=a
340 RETURN
```

El programa BASIC concluye con unas pocas líneas con sentencias REM que contienen recomendaciones para el uso de la rutina.

```
1000 REM LET m2=any location
     within the record
1010 REM IF m1=0 then m2 points
     to the start of the record
     and m3 points to the end
1020 REM IF m1=1 then m2 did not
     point to a location inside
     the buffer
```

Se ofrece a continuación la traducción al lenguaje máquina de la versión en BASIC de la rutina START. El programa debe ser ensamblado a partir de la posición 60 570. En la columna de la izquierda se presentan las etiquetas desde LA hasta LE. Las direcciones de las variables denominadas en la versión BASIC como 'm1' hasta 'm4' son las siguientes:

m1 60 569

m2 60 135 y 60 136

m3 60565 y 60566

m4 60567 y 60568

Start

```
  1 REM START mnemonics to be
        assembled
 10 REM go
 20 REM org 50000 60570
 30 REM ld a,0
 40 REM ld (60569),a
 50 REM ld hl,(60135)
 55 REM D
 60 REM ld bc,35000
 70 REM and a
 80 REM abc hl,bc
 90 REM jp z,A
100 REM jp m,B
110 REM ld hl,(60135)
120 REM ld a,(hl)
130 REM sub 139
140 REM jp z,C
150 REM dec hl
160 REM ld (60135),hl
170 REM jp D
175 REM C
180 REM inc hl
190 REM ld (60135),hl
195 REM A
200 REM ld hl,(60135)
205 REM E
210 REM inc hl
220 REM ld (60565),hl
230 REM ld bc,(60567)
240 REM add a,a
250 REM sbc hl,bc
260 REM jp p,B
270 REM ld hl,(60565)
280 REM ld a,(hl)
290 REM sub 139
300 REM ret z
310 REM ld hl,(60565)
320 REM jp E
325 REM B
330 REM ld a,1
340 REM ld (60569),a
```

```
350 REM ret
360 REM finish
370 RANDOMIZE USR 58000
```

El método de grabación en cinta utilizado en el Capítulo 6 debe usarse también aquí. El comando necesario para grabar el programa ensamblado es el siguiente:

SAVE "START"CODE 50000, 76

Esta grabación deberá realizarse en la cinta que contiene las tres rutinas del Capítulo 6 y que se denominaron 'mc'. Cuando este juego de rutinas esté completo, las seis podrán considerarse como una nueva versión de 'mc' que puede utilizarse por SPECTEXT y SPECFILE.

DUMP coge el registro localizado por FIND y START y lo carga en el buffer de la pantalla (cadena de la línea 2). DUMP carga la información meticulosamente, teniendo en cuenta el formulario diseñado por el usuario. Para almacenar el programa se utilizan cuatro posiciones de memoria que se inicializan en las primeras líneas.

```
 1 REM BASIC version of DUMP,
       for demonstration only,
       not to be entered
10 LET LA=60
11 LET LB=170
12 LET LC=220
```

Las líneas 20 y 30 leen el número de campos y almacenan el valor en la variable 'm1'. A continuación, la variable 'm3' toma el valor de la posición del primer campo en el formulario. De nuevo aquí, la línea 50 puede parecer inútil, pero es necesaria porque retrocede para obtener una nueva versión del contenido de 'm3'.

```
20 LET a=PEEK 34997
30 LET m1=a
40 LET hl=34850
50 LET m3=hl
60 LET hl=m3
```

Las líneas 70 y 100 son una forma rara de decir en BASIC «carga en el par 'de' el valor contenido en la dirección hacia la que apunta el registro 'hl'». Después, el par 'hl' apuntará hacia la siguiente posición de la pantalla y este valor se almacena en 'm3'.

```
 70 LET d=PEEK hl
 80 LET hl=hl+1
 90 LET e=PEEK hl
100 LET hl=hl+1
110 LET m3=hl
```

En lenguaje máquina, si se altera el contenido de los registros 'd' o 'e', también se modifica el contenido del par 'de'. La línea 120 es una forma de decir en BASIC 'ld, hl, de'. Las etiquetas de los campos ocupan los primeros nueve caracteres de las líneas de la pantalla, por ello, las líneas desde la 130 a la 150 hacen que el par 'de' tome el valor del primer espacio libre de la pantalla. El par 'hl' toma el valor de la dirección del siguiente carácter en el registro.

```
120 LET hl=d*256+e
130 LET bc=9
140 LET hl=hl+bc
150 LET de=hl
160 LET hl=m2
```

El programa hace ahora un bucle con las cuatro líneas siguientes hasta que 'hl' apunte hacia un carácter cuyo código sea superior a 127. Este será el marcador de fin de campo. En cada ciclo del bucle se trasfiere un carácter desde la dirección contenida en el registro 'hl' hasta la dirección contenida en el registro 'de'. Este proceso se corresponde con la instrucción máquina 'ldi', que es una versión simple de la instrucción 'ldir' mencionada en el Capítulo 6. Esta instrucción hace lo mismo que 'ldir', excepto que no se repite. Se utiliza aquí para evitarnos tener que averiguar cuantos caracteres hay que trasladar en cada momento. El bucle continúa hasta que se encuentre un marcador.

```
170 LET a=PEEK hl
180 LET a=a-127
190 IF a>=0 THEN  GO TO LC
200 POKE de,PEEK hl
210 GO TO LB
```

La línea 220 trasfiere la dirección del carácter siguiente al marcador a 'm3'. El número de campos que quedan se introduce en el acumulador y se decrementa. Si el resultado fuera cero, el registro entero habría sido transferido y el control debe retornar al programa principal. En caso contrario, se almacena el número

de campos que quedan en 'm1' y se comienza un nuevo ciclo del bucle.

```
220 LET hl=hl+1
230 LET m2=hl
240 LET a=m1
250 LET a=a-1
260 IF a=0 THEN  RETURN
270 LET m1=a
280 GO TO LA
```

La única información que precisa la computadora para usar esta rutina es la dirección del principio del registro. Esta deberá introducirse en 'm2' en el programa en BASIC y en las posiciones 60 496 y 60 497 en el código máquina. Las posiciones de memoria que corresponden a las variables utilizadas en la versión BASIC del programa son las siguientes:

m1	60 495	número de campos
m2	60 496 y 60 497	siguiente carácter
m3	60 498 y 60 499	

Dump

```
  1 REM DUMP mnemonics to be
         assembled
 10 REM go
 20 REM org 50000 60500
 30 REM ld a,(34997)
 40 REM ld (60495),a
 50 REM ld hl,34850
 60 REM ld (60498),hl
 65 REM C
 70 REM ld hl,(60498)
 80 REM ld e,(hl)
 90 REM inc hl
100 REM ld d,(hl)
110 REM inc hl
120 REM ld (60498),hl
130 REM ld h,d
140 REM ld l,e
150 REM ld bc,9
160 REM adc hl,bc
170 REM ld d,h
180 REM ld e,l
190 REM ld hl,(60496)
```

```
195 REM B
2ØØ REM ld a,(hl)
21Ø REM sub 127
22Ø REM jp p,A
23Ø REM ldi
24Ø REM jp B
245 REM A
25Ø REM inc hl
26Ø REM ld (6Ø496),hl
27Ø REM ld a,(6Ø495)
28Ø REM sub 1
29Ø REM ret z
3ØØ REM ld (6Ø495),a
31Ø REM jp C
32Ø REM finish
33Ø RANDOMIZE USR 58ØØØ
```

El código ensamblado puede grabarse después de START con el comando:

SAVE "DUMP"CODE 5ØØØØ, 62

Sort

La última parte del código máquina es la más ambiciosa, la rutina de clasificación (Fig. 9.1). Hay diez variables que se corresponden con posiciones de memoria en la versión en lenguaje máquina del programa BASIC. Será necesario recurrir a la tabla 9.1 de vez en cuando.

Tabla 9.1

m1	6Ø 173 y 6Ø 174	registro siguiente
m2	6Ø 175 y 6Ø 176	segundo registro más próximo
m3	6Ø 177 y 6Ø 178	tercer registro más próximo
m4	6Ø 179	bandera de intercambio
m5	6Ø 18Ø	bandera de fin de fichero
m6	6Ø 181	contador de campos
m7 y m8		almacenes internos
m9	6Ø 182	número del campo
m1Ø	6Ø 183	bandera de error

El programa sigue el algoritmo de clasificación por burbujas con ligeras modificaciones que lo hacen un poco más sofisticado de lo habitual. En el caso de que los dos primeros elementos no estén en orden, se intercambian y se levanta una bandera (toma el

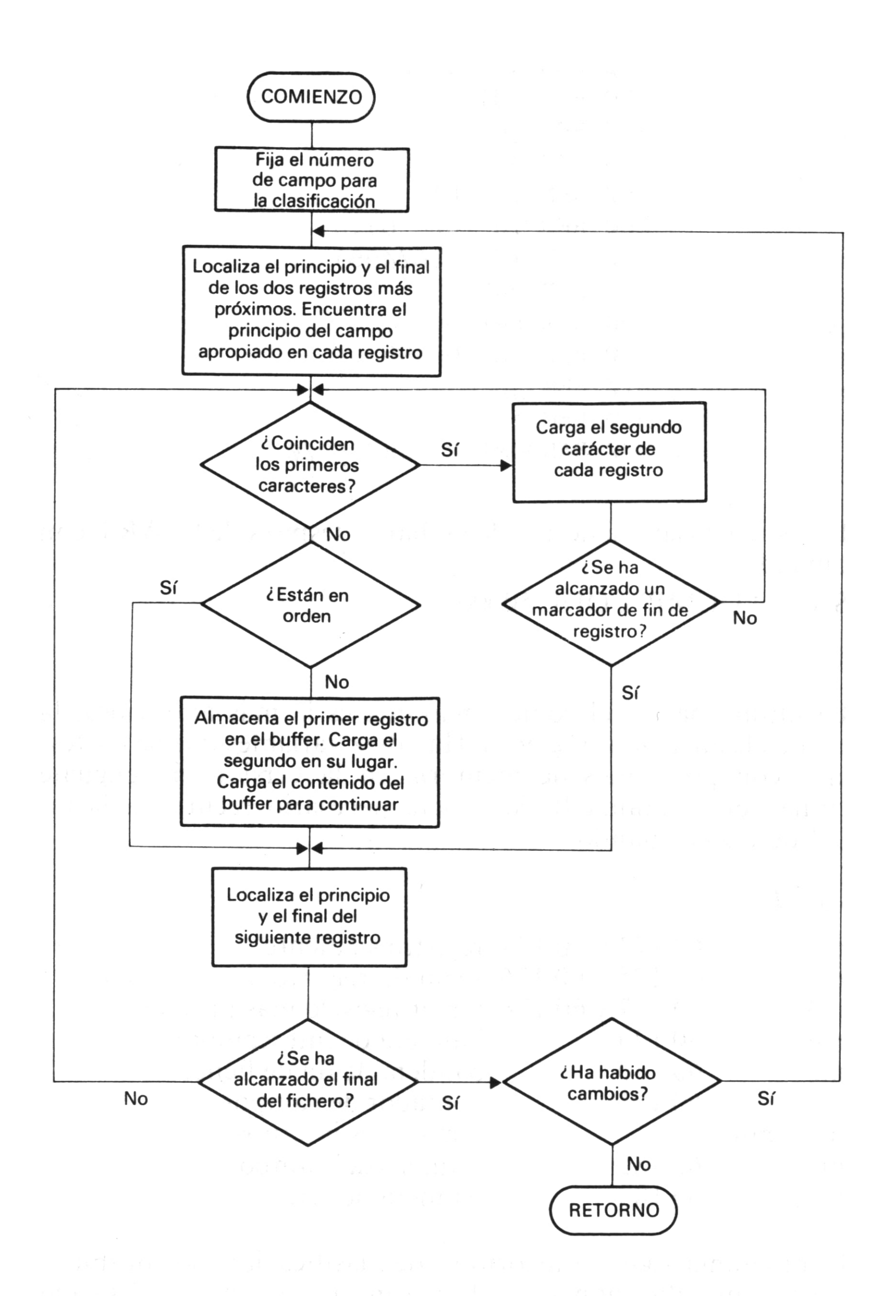

COMIENZO
Fija el número de campo para la clasificación
Localiza el principio y el final de los dos registros más próximos. Encuentra el principio del campo apropiado en cada registro
¿Coinciden los primeros caracteres?
Sí
Carga el segundo carácter de cada registro
No
¿Se ha alcanzado un marcador de fin de registro?
No
Sí
¿Están en orden
Sí
No
Almacena el primer registro en el buffer. Carga el segundo en su lugar. Carga el contenido del buffer para continuar
Localiza el principio y el final del siguiente registro
¿Se ha alcanzado el final del fichero?
No
Sí
¿Ha habido cambios?
Sí
No
RETORNO

Figura 9.1

valor uno). Después se comparan los elementos segundo y tercero y así sucesivamente hasta el final de la lista. Si la bandera tiene el valor uno al acabar el recorrido, significa que el fichero todavía no está ordenado. Unicamente cuando se alcance el final de la lista y la bandera no esté colocada significará que el fichero está ordenado. El método de intercambiar los registros se basa en la rutina MOVE de SPECTEXT. Pero pueden surgir algunos problemas que tenemos que tener en consideración. Si uno de los registros tuviera menos campos que otro, habría un error en el fichero que tendría que reseñarse. Si el fichero tuviera tan sólo un registro entonces no habría con que compararle.

Para poder intercambiar los registros es necesario conocer el principio y el final de los mismos. Para solucionar este problema SORT cuenta con una versión de la rutina START que permite la localización de los comienzos del registro siguiente y de los dos que continúan (segundo y tercero más próximos). Con esta información el programa puede intercambiar los registros sin problemas. Estos registros pueden ser bastante largos y por tanto si se utilizase el buffer de línea como zona de almacenamiento temporal de los registros, podría 'machacarse' la estructura del fichero. Por este motivo se utiliza como almacenamiento temporal para la rutina SORT la zona de memoria situada por encima del código máquina.

Antes de usar el programa la dirección del primer registro debe almacenarse en la posición 60 173 que corresponde a la variable 'm1' en el programa BASIC. Las primeras líneas del programa inicializan las banderas y las líneas 30 y 80 comienzan la búsqueda de un marcador.

```
 1 REM BASIC version of SORT,
       for demonstration only,
       not to be entered
10 LET a=0
15 LET m5=a
20 LET m4=a
30 LET hl=m1
40 LET a=PEEK hl
50 LET a=a-254
60 IF a=0 THEN  GO TO 90
70 LET hl=hl+1
80 GO TO 40
```

La dirección del byte siguiente al que contiene el marcador corresponde al principio del siguiente registro; la línea 100 almacena esta dirección en 'm2'. El próximo paso es verificar si se ha

alcanzado el marcador de final de fichero, en cuyo caso, el fichero contendría un solo registro y la clasificación sería imposible. La línea 72Ø almacena un mensaje de error en una variable antes de volver al programa principal. El mensaje es 'error 2, not enough records to sort'.

```
 9Ø LET hl=hl+1
1ØØ LET m2=hl
11Ø LET a=PEEK hl
12Ø LET a=a-255
13Ø IF a=Ø THEN  GO TO 72Ø
```

Si se supera el último obstáculo, la computadora comenzará a buscar un marcador de final de registro. Cuando se localice uno la línea 2ØØ archivará la dirección del comienzo de este segundo registro más próximo en 'm3'.

```
14Ø LET a=PEEK hl
15Ø LET a=a-254
16Ø IF a=Ø THEN  GO TO 185
17Ø LET hl=hl+1
18Ø GO TO 14Ø
19Ø LET hl=hl+1
2ØØ LET m3=hl
```

De nuevo hay que verificar si se ha alcanzado el final del fichero para colocar en su caso la bandera correspondiente.

```
21Ø LET a=PEEK hl
22Ø LET a=a-255
23Ø IF a<>Ø THEN  GO TO 26Ø
24Ø LET a=1
25Ø LET m5=a
```

Al final del programa hay una serie de controles cuya función es la de asegurarse que cada registro contiene el número exacto de campos y que los marcadores están posicionados en el orden correcto. Las estructuras de datos basadas en los marcadores, deben disponer de un conjunto completo y ordenado de los mismos para que el sistema de procesamiento de datos funcione adecuadamente. Los marcadores son la 'correa de trasmisión' del sistema. Con SORT es posible efectuar la clasificación por cualquiera de los campos del registro. La línea 26Ø llama a dos rutinas, una de veri-

ficación y otra que hace recorrer el registro a un puntero temporal hasta localizar el campo correcto. La línea 261 devuelve el control al programa principal si se encontrasen errores. La dirección correspondiente al principio del siguiente registro se almacena en 'm7' y se localiza el campo correcto. A continuación se prosigue con la misma rutina de verificación, haciéndola actuar sobre el segundo registro más próximo.

```
260 LET hl=m1: IF m9>1 THEN  GO
SUB 902
261 IF m10<>0 THEN  RETURN
262 LET m7=hl
263 LET hl=m2: IF m9>1 THEN  GO
SUB 902
264 IF m10<>0 THEN  RETURN
```

Las líneas comprendidas entre la 265 y la 320 comparan los primeros caracteres de cada registro. Si coincidieran, la computadora iría a la línea 750, donde se compararían los siguientes caracteres. El proceso continuará hasta que se encuentre una marca de fin de registro o una diferencia.

```
265 LET m8=hl
269 LET bc=m7
270 LET de=m8
280 LET a=PEEK bc
290 LET l=a
300 LET a=PEEK de
310 LET a=a-l
320 IF>a=0 THEN  GO TO 750
```

Las próximas cuatro líneas efectúan la comprobación final. Si el orden de los registros fuese correcto se saltaría la sección siguiente. En caso contrario se aplicaría una rutina de intercambio.

```
330 LET a=PEEK bc
340 LET l=a
350 LET a=PEEK de
360 LET a=a-l
370 IF a>=0 THEN  GO TO 608
```

Las primeras dos líneas de la rutina de intercambio son las encargadas de colocar la bandera correspondiente, 'm4'. Se calcula el número de bytes que hay que trasladar y se almacena en el par

de registros 'bc'. El destino temporal de los registros que van a ser intercambiados es el espacio de memoria situado por encima del código máquina (posición 62 ØØØ). El par de registros 'de' toma el valor de esta dirección y el 'hl' el de la dirección contenida en 'm1', que corresponde al principio del siguiente registro. Todo está preparado para la instrucción 'ldir' efectúe el movimiento, no obstante, el número de bytes que van a ser trasladados se empuja en la pila para que no sea preciso volver a calcularlo después. Sólo hay una instrucción PUSH y otra POP en este programa, por ello no se utiliza la simulación completa en BASIC de la pila de datos. La línea 47Ø simula la instrucción 'ldir'.

```
 38Ø LET a=1
 39Ø LET m4=a
 4ØØ LET hl=m2
 41Ø LET bc=m1
 42Ø LET hl=hl-bc
 43Ø LET bc=hl
 44Ø LET de=62ØØØ
 45Ø LET hl=m1
 46Ø LET bc1=bc
 47Ø FOR j=Ø TO bc-1: POKE de+j,
PEEK (hl+j): NEXT j: REM LDIR
```

En las líneas comprendidas entre la 48Ø y la 54Ø se cambian los punteros para hacer que el segundo registro más próximo se traslade al espacio dejado vacante por el siguiente registro.

```
 48Ø LET hl=m3
 49Ø LET bc=m2
 5ØØ LET hl=hl-bc
 51Ø LET bc=hl
 52Ø LET de=m1
 53Ø LET hl=m2
 54Ø FOR j=Ø TO bc-1: POKE de+j,
PEEK (hl+j): NEXT j: LET de=de+b
c: REM LDIR
```

La recuperación del número almacenado en 'bc1' es equivalente a la instrucción máquina POP. Después de la línea 55Ø, el par 'bc' contiene el número de caracteres del registro que ahora está almacenado en la posición 62 ØØØ, Se necesita un poco más de preparación antes de que vuelva a utilizarse la simulación de 'ldir', ahora para trasladar el antiguo 'siguiente' registro a conti-

nuación del 'segundo registro más próximo'. Los registros se han intercambiado.

```
 55Ø LET bc=bc1
 56Ø LET hl=62ØØØ
 57Ø FOR j=Ø TO bc-1: POKE de+j,
PEEK (hl+j): NEXT j: REM LDIR
```

Se calcula la dirección del principio del registro que acaba de trasladarse y después se actualizan 'm1' y 'm2'.

```
58Ø LET hl=m3
585 LET bc=m2
59Ø LET hl=hl-bc
592 LET hl=hl+m1
595 LET m1=hl
6ØØ LET hl=m3
6Ø5 LET m2=hl
6Ø6 GO TO 62Ø
```

Si los registros no han sido intercambiados, no es necesario calcular su nueva dirección inicial: 'm1' pasa a ser 'm2' y 'm2' pasa a ser 'm3'. Las líneas comprendidas entre la 6Ø8 y la 614 se encargan de modificar estas variables.

```
6Ø8 LET hl=m2
61Ø LET m1=hl
612 LET hl=m3
614 LET m2=hl
```

Si el valor del acumulador se suma a sí mismo (se multiplica por dos), el resultado será cero si, y sólo si, el valor inicial del acumulador fuera cero. De esta forma, las líneas comprendidas entre la 62Ø y la 64Ø comprueban la colocación de la bandera 'm5' indicadora del final del fichero. En el caso de que no haya sido alcanzado el final del fichero, la línea 65Ø enviará la computadora a efectuar un nuevo ciclo del bucle. Si la bandera hubiera sido colocada nos encontraríamos ante el último registro del fichero.

```
62Ø LET a=m5
63Ø LET a=a+a
64Ø IF a<>Ø THEN  GO TO 66Ø
645 LET hl=m2
65Ø GO TO 14Ø
```

Ahora, se verificará la bandera indicadora de intercambio usando el método recién descrito. Si no hubiera habido intercambios y se hubiese alcanzado el final del fichero, el mismo estaría ordenado y se volvería al programa principal. En caso contrario, el proceso ha de repetirse completamente y el programa tendrá que volver a la línea 1Ø.

```
66Ø LET a=m4
67Ø LET a=a+a
68Ø IF a=Ø THEN  RETURN
69Ø LET hl=35ØØØ
7ØØ LET m1=hl
71Ø GO TO 1Ø
```

A continuación se ofrecen rutinas que efectúan trabajos que han sido propuestos y dejados de efectuar anteriormente. Cada una de ellas va precedida de una pequeña nota explicativa de su función. La primera coloca a dos una bandera indicadora de final de fichero. Lo que significa que hay insuficientes registros en el fichero.

```
72Ø LET a=2
73Ø LET m5=a
74Ø RETURN
```

La siguiente rutina se encarga de la comparación del segundo bloque de caracteres.

```
75Ø LET bc=bc+1
76Ø LET de=de+1
77Ø LET a=PEEK bc
78Ø LET a=a-254
79Ø IF a=Ø THEN  GO TO 58Ø
8ØØ LET a=PEEK de
81Ø LET a=a-254
82Ø IF a=Ø THEN  GO TO 58Ø
83Ø GO TO 28Ø
```

El resto del programa consiste en un extenso bucle que consta de varias secciones pequeñas. El propósito del bucle es el de identificar errores y localizar el campo correcto dentro de un registro.

```
9Ø2 LET m6=m9
9Ø3 LET m6=m6-1
```

```
905 LET a=PEEK hl
910 LET a=a-254
915 IF a<>0 THEN  GO TO 920
916 LET a=3
917 LET m10=a
918 RETURN
```

La siguiente verificación se hace para saber si hay un marcador de fin de fichero donde debiera haber un marcador de fin de registro.

```
920 LET a=PEEK hl
925 LET a=a-255
930 IF a<>0 THEN  GO TO 935
931 LET a=4
932 LET m10=a
933 RETURN
```

Si se encuentra un marcador de fin de campo, la computadora comprueba si dicho campo es el correcto.

```
935 LET a=PEEK hl
940 LET a=a-253
945 IF a=0 THEN  GO TO 960
950 LET hl=hl+1
955 GO TO 905
```

Si el registro no tiene el número correcto de campos, es necesaria la línea 916.

```
960 LET hl=hl+1
962 LET a=PEEK hl
964 LET a=a-254
966 IF a=0 THEN  GO TO 916
```

El número de campo se decrementa antes de enviar a la computadora a la línea 905 para examinar nuevos campos. Si el número del campo llega a valer cero, se ha localizado el campo correcto y puede efectuarse la comparación.

```
970 LET a=m6
975 LET a=a-1
980 LET m6=a
```

```
985 IF a=Ø THEN  RETURN
988 GO TO 9Ø5
```

Aquí termina la parte más compleja del libro. Los números de línea se han descuidado deliberadamente, así pueden apreciarse las correcciones que fueron necesarias durante el proceso de desarrollo del programa. La traducción del programa BASIC a lenguaje máquina se llevó a cabo con pocos problemas. He aquí la traducción:

```
  1 REM SORT mnemonics to be
        assembled
 1Ø REM go
 2Ø REM org 5ØØØØ 6Ø185
 25 REM L
 3Ø REM ld a,Ø
 4Ø REM ld (6Ø18Ø),a
 5Ø REM ld (6Ø179),a
 6Ø REM ld hl,(6Ø173)
 65 REM A
 7Ø REM ld a,(hl)
 8Ø REM sub 139
 9Ø REM jp z,W
1ØØ REM inc hl
11Ø REM jp A
115 REM W
12Ø REM inc hl
13Ø REM ld (6Ø175),hl
14Ø REM ld a,(hl)
15Ø REM sub 143
16Ø REM jp z,B
165 REM D
17Ø REM ld a,(hl)
18Ø REM sub 139
19Ø REM jp z,C
2ØØ REM inc hl
21Ø REM jp D
215 REM C
22Ø REM inc hl
23Ø REM ld (6Ø177),hl
24Ø REM ld a,(hl)
25Ø REM sub 143
26Ø REM jp nz,E
27Ø REM ld a,1
```

```
28Ø REM ld (6Ø18Ø),a
285 REM E
29Ø REM ld hl,(6Ø173)
3ØØ REM ld a,(6Ø182)
31Ø REM dec a
32Ø REM call nz,F
33Ø REM ld a,(6Ø183)
34Ø REM add a,a
35Ø REM ret nz
36Ø REM ld (6Ø147),hl
37Ø REM ld hl,(6Ø175)
38Ø REM ld a,(6Ø182)
39Ø REM dec a
4ØØ REM call nz,F
41Ø REM ld a,(6Ø183)
42Ø REM add a,a
43Ø REM ret nz
44Ø REM ld (6Ø149),hl
45Ø REM ld bc,(6Ø147)
46Ø REM ld de,(6Ø149)
465 REM N
47Ø REM ld a,(bc)
48Ø REM ld l,a
49Ø REM ld a,(de)
5ØØ REM sub l
51Ø REM jp z,G
52Ø REM jp p,H
53Ø REM ld a,l
54Ø REM ld (6Ø179),a
55Ø REM ld hl,(6Ø175)
56Ø REM ld bc,(6Ø173)
57Ø REM sbc hl,bc
58Ø REM ld b,h
59Ø REM ld c,l
6ØØ REM ld de,62ØØØ
61Ø REM ld hl,(6Ø173)
62Ø REM push bc
63Ø REM ldir
64Ø REM ld hl,(6Ø177)
65Ø REM ld bc,(6Ø175)
66Ø REM sbc hl,bc
67Ø REM ld b,h
68Ø REM ld c,l
69Ø REM ld de,(6Ø173)
```

```
 700 REM ld hl,(60175)
 710 REM ldir
 720 REM pop bc
 730 REM ld hl,62000
 740 REM ldir
 745 REM M
 750 REM ld hl,(60177)
 760 REM ld bc,(60175)
 770 REM sbc hl,bc
 780 REM ld bc,(60173)
 790 REM adc hl,bc
 800 REM ld (60173),hl
 810 REM ld hl,(60177)
 820 REM ld (60175),hl
 830 REM jp J
 835 REM H
 840 REM ld hl,(60175)
 850 REM ld (60173),hl
 860 REM ld hl,(60177)
 870 REM ld (60175),hl
 875 REM J
 880 REM ld a,(60180)
 890 REM add a,a
 900 REM jp nz,K
 910 REM ld hl,(60175)
 920 REM jp D
 925 REM K
 930 REM ld a,(60179)
 935 REM add a,a
 940 REM ret z
 950 REM ld hl,35000
 960 REM ld (60173),hl
 970 REM jp L
 975 REM B
 980 REM ld a,2
 990 REM ld (60180),a
1000 REM ret
1005 REM G
1010 REM inc bc
1020 REM inc de
1030 REM ld a,(bc)
1040 REM sub 139
1050 REM jp z,M
1060 REM ld a,(de)
```

```
1Ø7Ø REM sub 139
1Ø8Ø REM jp z,M
1Ø9Ø REM jp N
1Ø95 REM F
11ØØ REM ld a,(6Ø182)
111Ø REM dec a
112Ø REM ld (6Ø181),a
1125 REM T
113Ø REM ld a,(hl)
114Ø REM sub 139
115Ø REM jp nz,P
1155 REM U
116Ø REM ld a,3
117Ø REM ld (6Ø183),a
118Ø REM ret
1185 REM P
119Ø REM ld a,(hl)
12ØØ REM sub 143
121Ø REM jp nz,R
122Ø REM ld a,4
123Ø REM ld (6Ø183),a
124Ø REM ret
1245 REM R
125Ø REM ld a,(hl)
126Ø REM sub 131
127Ø REM jp z,S
128Ø REM inc hl
129Ø REM jp T
1295 REM S
13ØØ REM inc hl
131Ø REM ld a,(hl)
132Ø REM sub 139
133Ø REM jp z,U
134Ø REM ld a,(6Ø181)
135Ø REM dec a
136Ø REM ld (6Ø181),a
137Ø REM ret z
138Ø REM jp T
139Ø REM finish
14ØØ RANDOMIZE USR 58ØØØ
```

Después de ensamblar la rutina, hay que grabarla a continuación de START y DUMP mediante el comando:

SAVE "SORT"CODE 5ØØØØ, 31Ø

El código máquina está finalizado. La colección completa puede grabarse en cinta para su uso con SPECTEXT y SPECFILE, aunque sólo SPECFILE utilice las rutinas. Para introducir las rutinas en la computadora utilizar, uno a uno, los comandos siguientes:

1. LOAD “FILL”CODE 60151
2. LOAD “MOVE”CODE 60000
3. LOAD “FIND”CODE 60028
4. LOAD “START”CODE 60570
5. LOAD “DUMP”CODE 60500
6. LOAD “SORT”CODE 60185

Cuando se hayan cargado todas las rutinas, debe hacerse una nueva grabación del paquete total con:

SAVE “mc”CODE 60000, 650

Es recomendable hacer una copia de seguridad.

Conviene que, las dos partes que componen SPECFILE se carguen por separado. Por este motivo se ha preparado un pequeño programa de carga. Este programa dará el valor 33 999 a la variable del sistema RANTOP, así se protegen el código máquina y el contenido del buffer de pulsar por accidente NEW. El resto del programa se ocupa de la carga de las dos partes del SPECFILE.

```
  10 CLEAR 33999: PRINT AT 10,7;
"LOADING SPECFILE"'''TAB 3;"Two
more sections to load"
  20 LOAD "mc"CODE : PRINT TAB 3
;"One more section to load": LOA
D "file"
```

Una vez introducidas estas líneas, tienen que grabarse mediante el comando:

SAVE “specfile”LINE 1

Inmediatamente después, grabar el código máquina con el comando:

SAVE “mc”CODE 60000, 650

Terminada la grabación hay que parar el magnetófono, pero no rebobinar la cinta. A continuación, utilizando la cinta en que grabamos la parte en BASIC de SPECFILE, cargar el contenido

en la computadora para grabarlo después de los dos anteriores con el comando:

SAVE "file"LINE 3

Para cargar la grabación compuesta que acabamos de preparar en la computadora es suficiente con la instrucción LOAD " ". De nuevo recomendamos que se realice una copia de seguridad de todo el programa.

Para aquellos lectores que no dispongan de un ensamblador, damos a continuación los bloques de código numérico que deben incluirse en instrucciones DATA y cargarse después utilizando la técnica del bucle FOR... NEXT descrita en el Capítulo 6.

Start

```
Numeric code for START
FOR j=6Ø57Ø TO 6Ø645

62    Ø     5Ø    153   236   42
231   234   1     184   136   167
237   66    2Ø2   194   236   25Ø
224   236   42    231   234   126
214   139   2Ø2   19Ø   236   43
34    231   234   195   162   236
35    34    231   234   42    231
234   35    34    149   236   237
75    151   236   135   237   66
242   224   236   42    149   236
126   214   139   2ØØ   42    149
236   195   197   236   62    1
5Ø    153   236   2Ø1
```

El siguiente programa carga START en su sitio.

```
1Ø  DATA  62, Ø, 5Ø, 153, 236, 42, 231, 234, 1, 184, 136,
          167, 237, 66, 2Ø2, 194, 236, 25Ø
11  DATA  224, 236, 42, 231, 234, 126, 214, 139, 2Ø2, 19Ø,
          236, 43, 34, 231, 234, 195, 162, 236
12  DATA  35, 34, 231, 234, 42, 231, 234, 35, 34, 149, 236,
          237, 75, 151, 236, 135, 237, 66
13  DATA  242, 224, 236, 42, 149, 236, 126, 214, 139, 2ØØ,
          42, 149, 236, 195, 197, 236, 62, 1, 5Ø, 153,
          236, 2Ø1
2Ø  FOR j = 6Ø57Ø TO 6Ø645: READ a: POKE j, a: NEXT j
```

Grabar este programa en BASIC y después ejecutarlo mediante RUN, así se situará el código en su posición correcta. Repetir el

proceso para las dos rutinas siguientes. El bucle FOR... NEXT tendrá que modificarse para adaptarse a cada caso.

Dump

```
Numeric code for DUMP
FOR j=6Ø5ØØ TO 6Ø56Ø

58   181  136  5Ø   79   236
33   34   136  34   82   236
42   82   236  94   35   86
35   34   82   236  98   1Ø7
1    9    Ø    237  74   84
93   42   8Ø   236  126  214
127  242  129  236  237  16Ø
195  118  236  35   34   8Ø
236  58   79   236  214  1
2ØØ  5Ø   79   236  195  96
236
```

Sort

```
Numeric code for SORT
FOR j=6Ø185 TO 6Ø493

62   Ø    5Ø   2Ø   235  5Ø
19   235  42   13   235  126
214  139  2Ø2  46   235  35
195  36   235  35   34   15
235  126  214  143  2Ø2  252
235  126  214  139  2Ø2  66
235  35   195  56   235  35
34   17   235  126  214  143
194  81   235  62   1    5Ø
2Ø   235  42   13   235  58
22   235  61   196  19   236
58   23   235  135  192  34
243  234  42   15   235  58
22   235  61   196  19   236
58   23   235  135  192  34
245  234  237  75   243  234
237  91   245  234  1Ø   111
26   149  2Ø2  2    236  242
213  235  62   1    5Ø   19
235  42   15   235  237  75
13   235  237  66   68   77
17   48   242  42   13   235
197  237  176  42   17   235
237  75   15   235  237  66
```

68	77	237	91	13	235
42	15	235	237	176	193
33	48	242	237	176	42
17	235	237	75	15	235
237	66	237	75	13	235
237	74	34	13	235	42
17	235	34	15	235	195
225	235	42	15	235	34
13	235	42	17	235	34
15	235	58	2Ø	235	135
194	238	235	42	15	235
195	56	235	58	19	235
135	2ØØ	33	184	136	34
13	235	195	25	235	62
2	5Ø	2Ø	235	2Ø1	3
19	1Ø	214	139	2Ø2	186
235	26	214	139	2Ø2	186
235	195	125	235	58	22
235	61	5Ø	21	235	126
214	139	194	38	236	62
3	5Ø	23	235	2Ø1	126
214	143	194	5Ø	236	62
4	5Ø	23	235	2Ø1	126
214	131	2Ø2	6Ø	236	35
195	26	236	35	126	214
139	2Ø2	32	236	58	21
235	61	5Ø	21	235	2ØØ
195	26	236			

Una vez que los tres programas BASIC se hayan ejecutado, el código máquina estará en su lugar. Cogeremos la cinta que contenga el código máquina de SPECTEXT y cargaremos el código mediante la instrucción:

LOAD "mc"CODE

Las seis rutinas están ahora en memoria entre las posiciones 6Ø ØØØ y 6Ø 65Ø. Grabar el paquete completo con:

SAVE "mc"CODE 6Ø ØØØ, 65Ø

Vuelva a las páginas 128-129 para construir el paquete final de SPECFILE comenzando con el programa de carga.

10 SPECMERGE

Este programa está destinado a los lectores que, por diversas razones, necesiten escribir habitualmente gran número de cartas o documentos de tipo estándar, pero cada uno de ellos adaptado a su destinatario.

Pongámonos en el lugar de un joven que acaba de recibir una computadora como regalo de cumpleaños. Con la computadora puede jugar a los 'marcianitos' gratis durante horas y horas. No tiene tiempo de escribir cartas de agradecimiento pero, por otra parte, estas cartas son necesarias para asegurarse los regalos del próximo año. La solución está en la base del problema, la computadora. Las cartas de agradecimiento son difíciles de escribir, porque aparte de dar las gracias hay poco más que decir y, sin embargo, eso no es suficiente. La carta tiene que repetirse, además, tantas veces como personas tengan que recibirla. Las computadoras son un buen instrumento para realizar aquellas tareas para las que la gente no está dispuesta y este es un buen ejemplo. El usuario se encarga de la parte creativa, escribir la carta matriz. La computadora realizará la parte monótona utilizando la carta matriz y un fichero.

Una vez escrita la carta, es preciso incluir una serie de marcadores que señalarán los puntos en los que es necesario introducir algún cambio. El destinatario, la dirección, el regalo y algunas otras partes de la carta tendrán que modificarse. Puede usarse SPECTEXT para escribir la carta, incluyendo marcadores allí donde los datos deban buscarse en el fichero. SPECFILE podría usarse para crear el fichero que contuviera los detalles individualizantes. El nuevo programa SPECMERGE escribirá la carta hasta encontrar un marcador, después consultará el fichero para conocer el texto específico a insertar. A continuación la computadora continuará escribiendo la parte estándar de la carta. Supongamos que tuviéramos que escribir una serie de cartas a familiares y amigos agradeciendo los regalos recibidos en el cumpleaños. Las partes variables de la carta serían los campos de los registros del fichero y podrían ser los siguientes:

1. TITULO, Sr., Sra., Srta., etc.

2. NOMBRE

3. APELLIDO
4. CALLE
5. NUMERO
6. POBLACION
7. PROVINCIA
8. PARENTESCO, tío, tía, abuelo, etc.
9. REGALO
10. RELACION, sobrino, nieto, etc.

El marcador deberá ser una señal que nunca se vaya a emplear en las cartas, por ejemplo, una flecha '↑'. El programa SPECMERGE al encontrar estas flechas lee el contenido del campo a que se refiere cada una de ellas. Es importante que los números de los campos entre las flechas, se separen entre sí mediante comas. En el ejemplo, el campo 9 es en el que se escribe el regalo recibido. La carta matriz se escribiría en inglés así: 'Thank you for the ↑9↑ which you sent me for my birthday' y la carta impresa para ser dirigida tendría el texto siguiente: 'Thank you for the cycling gloves which you...' o bien 'Thank you for the interesting book which you...' dependiendo de lo que estuviera almacenado en el campo 9 en cada uno de los distintos registros.

A continuación se da un ejemplo en inglés de carta matriz que podría utilizar SPECMERGE.

```
↑1,2,3↑▬↑4,5↑▬↑6↑▬↑7↑▬▬▬Dear ↑8,
2↑,▬Thank you very much indeed f
or the ↑9↑ that you sent me for
my birthday. It was very kind of
 you to remember me.▬I am writin
g this letter on the computer wh
ich Mum and Dad gave me for my b
irthday. Thanks, once again for
the ↑9↑.▬Your ↑10↑,▬     Jago.
```

y el fichero creado con SPECFILE podría ser:

```
Miss ▬Matilda ▬Smallperson ▬The
Cottage ▬Mill Lane ▬Nutford ▬Rut
land ▬Auntie ▬pair of socks ▬aff
ectionate nephew ▬▬Mr ▬Bert ▬Scr
oggins ▬1 ▬Piggery Street ▬Nutfo
rd ▬Rutland ▬Uncle ▬ferret ▬mate
 ▬▬Miss ▬Jane ▬Smallperson ▬15 ▬
Long Street ▬London ▬N599 ▬ ▬woo
lly jumper ▬ever loving brother
▬▬Mr ▬Scrooge ▬Meanbody ▬27 ▬Bac
k Lane ▬East Nutfield ▬Rutland ▬
```

```
Uncle ■good wishes ■nephew ■■Dam
e ■Gertrude ■Beneficient ■The Gr
ange ■ ■Nutfield ■Rutland ■Great
 Aunt ■£50 ■grateful nephew ■■
```

El programa necesario para fusionar la carta matriz y el fichero con el que obtener las cartas es muy sencillo y casi una tercera parte es prácticamente una copia de la rutina de formateo de SPECTEXT. Las cartas que se escribirán con los datos anteriores son:

```
Miss Matilda Smallperson
The Cottage Mill Lane
Nutford
Rutland

Dear Auntie Matilda,
     Thank you very much indeed
for the pair of socks that you
sent me for my birthday. It was
very kind of you to remember me.
     I am writing this letter on
the computer which Mum and Dad
gave me for my birthday. Thanks,
once again for the pair of
socks.
     Your affectionate nephew,
     Jago


Mr Bert Scroggins
1 Piggery Street
Nutford
Rutland

Dear Uncle Bert,
     Thank you very much indeed
for the ferret that you sent me
for my birthday. It was very
kind of you to remember me.
     I am writing this letter on
the computer which Mum and Dad
gave me for my birthday. Thanks,
once again for the ferret.
     Your mate,
     Jago
```

Miss Jane Smallperson
15 Long Street
London
NS99
Dear Jane,
Thank you very much indeed for the woolly jumper that you sent me for my birthday. It was very kind of you to remember me.
I am writing this letter on the computer which Mum and Dad gave me for my birthday. Thanks, once again for the woolly jumper.
Your ever loving brother,
Jago

Mr Scrooge Meanbody
27 Back Lane
East Nutfield
Rutland

Dear Uncle Scrooge,
Thank you very much indeed for the good wishes that you sent me for my birthday. It was very kind of you to remember me.
I am writing this letter on the computer which Mum and Dad gave me for my birthday. Thanks, once again for the good wishes.
Your nephew,
Jago

Dame Gertrude Beneficient
The Grange
Nutfield
Rutland

Dear Great Aunt Gertrude,
Thank you very much indeed for the £5Ø that you sent me for my birthday. It was very kind of you to remember me.
I am writing this letter on the computer which Mun and Dad gave me for my birthday. Thanks, once again for the £5Ø.
Your grateful nephew,
Jago

SPECMERGE se ha escrito casi completamente en BASIC. La rutina MOVE se utiliza para trasladar la carta en la memoria antes de cargar el fichero desde la cinta. Puesto que no es necesario introducir la totalidad del código de MOVE para este trabajo, se incluye parte de dicho código en una línea DATA del programa. La mayor parte del espacio de almacenamiento está dividido en dos partes, una de las cuales ocupa las direcciones superiores a 61 000 y la otra las posiciones comprendidas entre la 29 500 y la 34 000. Ambas áreas contienen alrededor de 4500 caracteres, lo que permite cartas bastante largas. En la primera zona se almacena la carta matriz (con sus marcadores) y la segunda contendrá las diferentes versiones de la carta antes de su impresión. El programa localizará primeramente los campos dentro de cada registro y almacenará las direcciones en una tabla. Una vez que los campos hayan sido localizados, la carta se copiará carácter a carácter en la segunda zona de almacenamiento hasta encontrar un marcador. La computadora buscará entonces el número de campo que sigue al marcador y copiará éste en la nueva carta. Si hubiera una coma después del número del campo, el próximo se copiaría a continuación. Así sucesivamente hasta que la computadora encontrase otro marcador. El segundo marcador indica a la máquina que continúe copiando la carta desde el punto en que se paró. Cuando se termine de copiar la carta, se preparará y escribirá sobre papel mediante la rutina de impresión y formateo. Hay que inicializar una serie de variables antes de cargar el código máquina por medio del bucle de la línea 10. La instrucción DATA de la línea 11 contiene el código máquina de la rutina MOVE.

```
  1Ø LET q=2367Ø: LET r=23671: R
ESTORE : FOR j=6ØØØØ TO 6ØØ13: R
EAD k: POKE j,k: NEXT j: GO TO 1
ØØØ
  11 DATA 1,Ø,Ø,33,Ø,Ø,84,93,33,
Ø,Ø,237,176,2Ø1
```

Sólo se utilizan tres subrutinas en el programa, dos de las cuales son copias directas de otras dos subrutinas de SPECTEXT. La rutina (20) imprime y cuenta las líneas formateadas. Cuando la pantalla esté llena puede realizarse la copia. La rutina (30) prepara y después llama a la rutina MOVE.

```
  2Ø PRINT b$( TO y-x): LET x=y+
1
  21 IF x1=22 AND p=1 THEN  COPY
```

```
: CLS : LET x1=Ø
  22 IF x1=22 AND p=Ø THEN  INPU
T c$: CLS : LET x1=Ø
  23 LET x1=x1+1: LET x4=x4+1: R
ETURN
  3Ø RANDOMIZE v1: POKE 6ØØØ1,PE
EK q: POKE 6ØØØ2,PEEK r: RANDOMI
ZE v2: POKE 6ØØØ4,PEEK q: POKE 6
ØØØ5,PEEK r: RANDOMIZE v3: POKE
6ØØØ9,PEEK q: POKE 6ØØ1Ø,PEEK r:
 RANDOMIZE USR 6ØØØØ: RETURN
```

La rutina (7Ø) se asegura de que los nombres de los ficheros tengan exactamente 10 caracteres de longitud y después solicita la confirmación del usuario.

```
  7Ø IF LEN z$>1Ø THEN  LET b$="
"
  71 IF LEN z$<>1Ø THEN  LET z$=
z$+" ": GO TO 71
  72 INPUT "Key C if OK ";b$: RE
TURN
```

Este programa no utiliza menú de opciones y comienza a ejecutarse en la línea 1ØØØ, en lugar de la línea 9ØØ como venía siendo habitual. La carta tiene que cargarse desde cinta, finalizada la carga la longitud puede leerse en las posiciones 34 998 y 34 999. La carta contiene un marcador (CHR$ 143) que señala el final de la misma e indica a la computadora que está lista para el proceso de formateo. En esta etapa se rechazarán las cartas que sobrepasen el espacio disponible (línea 1Ø3Ø).

```
1ØØØ CLS : PRINT TAB 1Ø;" SPECME
RGE "
1Ø1Ø PRINT '''Type the leter nam
e ": INPUT z$: PRINT ''z$: GO SU
B 7Ø: IF b$<>"c" OR z$="" THEN
GO TO 1
1Ø2Ø PRINT '''Start the tape": L
OAD z$CODE : CLS
1Ø3Ø LET c1=PEEK 34998+256*PEEK
34999: POKE 34997+c1,143: IF c1>
45ØØ THEN  PRINT "Letter too lon
g": PAUSE 25Ø: GO TO 1
```

Las líneas comprendidas entre la 1040 y la 1060 se han tomado de la rutina de mantenimiento de SPECTEXT. La bandera 'p' indica a la computadora si la impresora está conectada.

```
1040 CLS : PRINT "Use the printe
r?(y/n)": INPUT z$: IF z$<>"y" A
ND z$<>"n" THEN  GO TO 1040
1050 LET p=0: IF z$="y" THEN  LE
T p=1
1060 LET x3=80: IF p=1 THEN  PRI
NT "How many lines per page?": I
NPUT x3
```

La carta ocupa ahora el espacio destinado al fichero y por ello debe trasladarse a la dirección 61 000. La línea 1070 pide a la rutina (30) que recurra al código máquina para trasladar 'c1' bytes desde la posición 35 000 a la posición 61 000. Las líneas 1100 y 1110 cargan el fichero en el espacio vaciado para la ocasión.

```
1070 LET v1=c1: LET v2=61000: LE
T v3=35000: GO SUB 30
1100 CLS : PRINT "Type the file
name ": INPUT z$: PRINT ''z$: GO
 SUB 70: IF b$<>"c" OR z$="" THE
N  GO TO 1070
1110 PRINT ''"Start the tape": L
OAD z$CODE : CLS
```

Ahora que la carta y el fichero están en su lugar puede comenzar el verdadero trabajo. La línea 1120 lee en 'c5' la longitud del fichero y en 'a' el número de campos por registro. El puntero 'z' recorrerá el fichero hasta alcanzar el marcador de fin de fichero (CHR$ 143). No obstante, antes de que alcance el final del fichero tiene que realizarse gran cantidad de trabajo que en ocasiones pasa desapercibido. Los mensajes intermitentes vienen siendo utilizados para indicar que el sistema se mantiene en funcionamiento y que todo está bajo control. El primer trabajo a realizar es crear una tabla b(a+1) que contenga las direcciones de los campos dentro de los registros.

```
1120 LET c5=PEEK 34998+256*PEEK
34999: LET a=PEEK 34997
1130 LET z=35000
1140 CLS : PRINT  FLASH 1;"Build
```

```
ing next letter": DIM b(a+1): IF
 PEEK z+143 THEN  GO TO 5ØØØ
```

SPECFILE marca el final de un campo con CHR$ 131 y el final de un registro con CHR$ 139. El final del fichero se señala con CHR$ 143. La dirección del primer byte del primer campo será el valor de 'z', por tanto pasa a ser el primer elemento de la tabla b(1). A continuación se entrará en un bucle para localizar el marcador CHR$ 131. Al finalizar el bucle la tabla contendrá suficiente información para permitir a la computadora localizar cualquier campo dentro del registro. La línea 119Ø se encarga de informar al usuario.

```
115Ø LET b(1)=z: FOR j=1 TO a
116Ø IF BEEK z=131 THEN  LET b(j
+1)=z+1: GO TO 118Ø
117Ø LET z=z+1: GO TO 116Ø
118Ø LET z=z+1: NEXT j
119Ø PRINT "Fields found"
```

Un nuevo puntero 'w' recorre la carta matriz desde la posición 61 ØØØ y otro 'x' recorre el fichero desde la posición 29 5ØØ. El marcador '↑' tiene el código 94 y es buscado por la línea 121Ø. El resto de los caracteres son copiados de un área a otra hasta que 'w' llega al final. (c1+61 ØØØ). Cuando esto ocurre las variables se preparan para la carta siguiente y después se llama a la rutina de la línea 3ØØØ para imprimir la anterior. La línea 124Ø envía a la computadora en busca de más caracteres hasta que se localice el último.

```
12ØØ LET w=61ØØØ: LET x=295ØØ
121Ø IF PEEK w=94 THEN  GO TO 2Ø
ØØ
122Ø POKE x,PEEK w: LET x=x+1: L
ET w=w+1: IF w=61ØØØ+c1 THEN  CL
S : GO SUB 3ØØØ: LET w=61ØØØ: LE
T z=z+1: GO TO 114Ø
124Ø GO TO 121Ø
```

Ahora que tiene que insertarse un campo, se lleva 'w' hasta la posición siguiente al marcador, de forma que pueda localizarse el número de campo. No es posible leer el número del campo utilizado por medio de 'VAL CHR$ PEEK w' debido a que podría haber 15 campos y algunos tendrían entonces dos dígitos. Las tres

líneas siguientes cargan los caracteres numéricos en b$ hasta que se encuentre una coma o un marcador.

```
2ØØØ LET w=w+1: LET b$=""
2Ø1Ø IF PEEK w=94 OR PEEK w=44 T
HEN  GO TO 2Ø3Ø
2Ø2Ø LET b$=b$+CHR$ PEEK w: LET
w=w+1: GO TO 2Ø1Ø
```

Si la carta está correctamente escrita, VAL b$ será el número del campo y el bucle de la línea 2Ø3Ø encontrará los caracteres de dicho campo. Después se insertarán estos caracteres en la carta y ‘x’ (puntero de la carta matriz) se desplaza a otra posición. Si se encontrase un marcador ‘↑’ entonces ‘w’ se avanzaría y ‘x’ se retrocedería una posición. El puntero ‘x’ retrocede para evitar que la carta puede llenarse de espacios. Si en lugar de un marcador se encontrase a continuación otro número de campo se recurriría a la línea 2Ø5Ø que hará retroceder a la computadora hasta la línea 2ØØØ para investigar de qué campo se trata.

```
2Ø3Ø LET f=VAL b$: FOR j=b(f) TO
 b(f+1)-2: POKE x,PEEK j: LET x=
x+1: NEXT j
2Ø4Ø IF PEEK w=94 THEN  LET w=w+
1: LET x=x-1: GO TO 121Ø
2Ø5Ø GO TO 2ØØØ
```

Este es todo el programa necesario para unir el fichero y la carta. Sin embargo, todavía hay que formatear y escribir las cartas; para ello se utiliza la rutina de SPECTEXT que se modifica levemente para adecuarla a nuestras necesidades.

```
3ØØØ LET x=295ØØ
3Ø1Ø LET x1=1: LET x2=x1: LET x4
=x1
3Ø2Ø LET y=31
3Ø3Ø LET f=PEEK x: IF f=144 THEN
  GO TO 3Ø5Ø
3Ø35 IF f=42 THEN  GO TO 4ØØØ
3Ø4Ø IF f>127 THEN  LET x=x+1: I
F f<>131 THEN  GO TO 32ØØ
3Ø5Ø IF INKEY$="z" THEN  GO TO 1
3Ø6Ø LET f=Ø: LET b$="": FOR j=x
 TO x+y: LET b$=b$+CHR$ PEEK j:
```

```
IF (PEEK j>127 AND PEEK j<>144)
AND f=Ø THEN  LET f=j
3Ø7Ø NEXT j: LET y=j: IF f>Ø THE
N  LET y=f
3Ø8Ø LET f=Ø: FOR j=x TO y: IF P
EEK j=32 THEN  LET f=1: LET j=y
3Ø9Ø NEXT j: IF f=Ø THEN  GO TO
311Ø
31ØØ IF PEEK y<>32 AND PEEK y<12
8 THEN  LET y=y-1: GO TO 31ØØ
311Ø GO SUB 2Ø: IF PEEK y>127 TH
EN  LET x=x-1
312Ø GO SUB 34ØØ: GO TO 3Ø2Ø
32ØØ IF f<>139 THEN  GO TO 33ØØ
321Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: LET k=x3-x4: IF k<3 THEN  F
OR j=1 TO k: PRINT : GO SUB 21:
GO SUB 34ØØ: NEXT j: GO TO 323Ø
322Ø PRINT : GO SUB 21: GO SUB 3
4ØØ: PRINT : GO SUB 21: GO SUB 3
4ØØ
323Ø PRINT TAB 5;: LET y=26: GO
TO 3Ø3Ø
33ØØ IF f<>134 THEN  GO TO 343Ø
331Ø LET k=x3-x4: PRINT : GO SUB
 21: FOR j=1 TO k: PRINT : GO SU
B 21: NEXT j: GO SUB 341Ø
332Ø GO TO 3Ø2Ø
34ØØ IF x4<x3 OR p=Ø THEN  RETUR
N
341Ø FOR j=1 TO 5: PRINT : GO SU
B 21: IF j=3 THEN  PRINT TAB 15;
x2: LET x2=x2+1: GO SUB 21
342Ø NEXT j: LET x4=1: RETURN
343Ø IF p=Ø THEN  FOR j=1 TO 5:
PRINT : NEXT j
344Ø IF p=1 THEN  COPY
345Ø LET z=z+1: GO TO 114Ø
```

La línea 5000 ofrece al usuario la posibilidad de producir una segunda copia de todas las cartas.

```
5ØØØ CLS : PRINT AT 9,9;"End of
List""""Key C for another copy"
```

```
: INPUT b$: IF b$="c" THEN  GO T
O 112Ø
```

Este corto programa saca el máximo partido de SPECTEXT y de SPECFILE. Si bien ha sido diseñado principalmente para confección de cartas como las que acabamos de ver, podría servir para una más amplia gama de utilizaciones. Este programa en BASIC ha sido ofrecido aquí como un ejemplo de los programas de integración. No obstante, es preciso reconocer que adolece de la velocidad, calidad y sofisticación precisa para que pueda ser utilizado en el mundo de los negocios.

En el próximo capítulo se verán muchas sugerencias para la aplicación de los programas. Se explica además la forma en que el software puede adaptarse para utilizar una impresora distinta de la ZX Printer, de este modo se podrán producir cartas en papel en blanco con la calidad exigida en el mundo de los negocios.

11 USOS DE ESTOS PROGRAMAS

Existen dos formas básicas de diseño de software. Los programas pueden diseñarse para satisfacer unas necesidades concretas, o bien para una aplicación de tipo general dentro de un campo específico. Aunque inicialmente podría parecer que los programas para un uso específico son los ideales, las desventajas aparecen cuando se pretenden introducir ligeras modificaciones para adaptarlos a nuevas necesidades. Por este motivo, esta colección de programas se ha escrito en base al segundo criterio. Por ejemplo, SPECFILE puede utilizarse en una gran variedad de situaciones. Este capítulo contiene una serie de sugerencias para la utilización de los programas del libro. La primera, muestra cómo un pequeño negocio puede utilizar SPECFILE para incrementar la eficacia de su gestión. Por medio de clasificaciones sucesivas se racionaliza la toma de decisiones sobre el calendario de trabajos.

TRABAJO PENDIENTE

El orden en que se lleva a cabo un trabajo es de suma importancia para una empresa. Muchos factores intervienen en la organización y ordenamiento de los trabajos, por ejemplo: lugar, importancia del cliente, disponibilidad de herramientas, tiempo de espera, etc.

La estructura del fichero que se ofrece a continuación contiene una selección de estos factores como nombres de campo. Cuando el fichero esté completo se podrá clasificar en base a estos campos. En el apartado operaciones con el fichero se dan ideas sobre la forma de llevar a cabo estas clasificaciones del modo más eficiente.

Estructura del fichero

Nombre del campo	Descripción
Cliente	Nombre, dirección y teléfono
Area	Lugar del trabajo
Entrada	Fecha en la que el trabajo entró en nuestra cartera

Prioridad	Importancia del cliente
Varios	Cualquier detalle de interés

Operaciones con el fichero

A medida que nos entren trabajos, se darán de alta registros en el fichero que se irán eliminando conforme se ultimen los trabajos correspondientes. Así el fichero constituye la cartera de trabajos pendientes. Puesto que los trabajadores de la compañía tienen que visitar a los diferentes clientes para solucionarles su problema, el fichero puede utilizarse para optimizar los desplazamientos. Para ello puede utilizarse la rutina de clasificación tantas veces como sea necesaria. Ahora bien, cuando se clasifica por diversos campos debe tenerse en cuenta que cada clasificación atenúa los efectos de la anterior. Si se estima que deben tenerse en cuenta varios campos para ordenar los trabajos, antes de proceder a las clasificaciones deben jerarquizarse por orden de importancia los atributos correspondientes a dichos campos. La primera clasificación debe realizarse por el campo menos importante de los que hay que tener en cuenta, la segunda por el siguiente en importancia y así sucesivamente hasta que la última clasificación (por el campo más importante) haya concluido.

Una organización de carácter nacional podría considerar que el 'área' es el factor más importante. Para un comerciante local probablemente la 'prioridad' sea el campo de mayor importancia. Supongamos que el campo más importante sea 'área' seguido de 'prioridad' y 'entrada'. La clasificación deberá realizarse primero por 'entrada', después por 'prioridad' y por último por 'área'. Mediante la impresora podría obtenerse así un calendario de trabajos para la semana. No obstante, convendría que de vez en cuando se efectuase una clasificación tan solo por el campo 'entrada' con el fin de identificar los pedidos que permanecen en la cartera desde hace tiempo.

LISTA DE CUMPLEAÑOS

En las grandes familias es difícil estar al tanto y controlar todos los regalos de cumpleaños, felicitaciones de Navidad, invitaciones de boda, etc. No es normal que alguien imprima sus invitaciones de boda con una computadora, pero sin embargo, la máquina podría escribir las etiquetas adhesivas de los sobres. La rutina de clasificación puede emplearse para verificar mensualmente las fechas de cumpleaños. Para ello es preciso diseñar un fichero que contenga para cada familiar la fecha de cumpleaños.

En primer lugar aparecerá el número del mes, posteriormente, el número del día. En los casos en que el mes tenga un solo dígito, éste será precedido por un cero. Los números de mes y día deberán separarse por el signo '/'. Por ejemplo, el 9 de mayo se escribirá 'Ø5/9'.

Estructura del fichero

Nombre del campo	Descripción
Título	Sr., Sra., Srta., etc.
Nombre	Nombre
Apellido	
Dirección	
Población	
País	
Teléfono	
FDN	Fecha de nacimiento

Operaciones con el fichero

Este fichero puede utilizarse como una agenda de direcciones para escribir sobres y cartas. SPECTEXT serviría para escribir la carta matriz y SPECMERGE para combinar el texto de la carta y el fichero y generar de este modo cartas individuales. Si se quiere escribir los destinatarios solo en los sobres, usar SPECTEXT para escribir una carta estándar.

CRIANZA DE PERROS

Los campeones no aparecen por casualidad, es necesaria una crianza cuidadosa para fortalecer las características de una raza. Sería provechoso disponer de un fichero de perros sementales a la hora de decidir los cruces a efectuar.

Estructura del fichero

Nombre del campo	Descripción
Perro	Nombre del perro
Dueño	Nombre del propietario, dirección y teléfono
Gastos	Comisión a pagar por el cruce

FDN	Fecha de nacimiento
1.	Característica 1
.	.
.	.
.	.
11.	Característica 11

Operaciones con el fichero

Preparar una lista de las características que se desean para las crías, por ejemplo: estructura ósea, tipo y calidad de pelambre, temperamento, etc. Hacer una clasificación de los machos en base a la menos importante de dichas cualidades, repitiendo la clasificación en orden ascendente de grado de importancia, hasta llegar a la característica más importante. Imprimir una lista de los sementales más adecuados y utilizar SPECTEXT para escribir una carta matriz y SPECMERGE para enviar una carta personal a cada uno de los propietarios de los perros que aparezcan en la lista de candidatos.

AGENCIA DE TEATRO AMATEUR

Los agentes de teatro y los comités de clubs de teatro se enfrentan a numerosos problemas. Quieran o no creerlo, esta es una excelente oportunidad para dejar a la computadora decidir la composición de un determinado elenco artístico. La estructura del fichero que nos serviría, podría ser la siguiente.

Estructura del fichero

Nombre del campo	Descripción
Nombre	
Dirección	
Edad	
Clásico	Valoración del talento en esta área
Comedia	
Moderno	
Arte	Capacidad para asumir papeles de protagonista
Canto	Tipo de voz y calidad
Danza	Estilo y aptitudes

Operaciones con el fichero

La codificación tiene que estar cuidadosamente organizada para que se pueda sacar un buen partido de este fichero. Consideremos, por ejemplo, el campo de la danza. Pueden codificarse estilos diferentes como zapateado y ballet para que después sea posible la clasificación. Si algún intérprete estuviera cualificado para el ballet y para el zapateado, sería necesaria la creación de una codificación que permitiera combinaciones. Podría utilizarse, con ligeras modificaciones, el método usado en la lista de cumpleaños. Nuevamente habría que clasificar en primer lugar por el campo menos importante y continuar como siempre hasta la última clasificación que tendría que hacerse por el campo más importante.

COLECCION DE CASSETTES

Actualmente incluso las colecciones privadas de cassettes pueden alcanzar estimables proporciones, siendo a veces difícil encontrar una grabación concreta. SPECFILE proporciona en este caso una ayuda considerable. El fichero contendrá las grabaciones en vez de las cassettes.

Estructura del fichero

Nombre del campo	Descripción
Título	Nombre de la grabación
Contenido	Breve descripción del tema grabado
Longitud	Duración de la grabación
Posición	Cara, número de revoluciones
Importancia	Valoración de la importancia para el propietario

Operaciones con el fichero

El primer uso que se nos ocurre es la localización de una grabación concreta. Si no se conoce exactamente el Nombre puede utilizarse en la búsqueda el campo Contenido. Si no dispusiéramos de cintas para realizar una grabación, podríamos seleccionar una grabación para registrar encima mediante la clasificación del fichero por el campo Importancia.

LISTA DE MIEMBROS DE UN CLUB

Este ejemplo es una sencilla variante del sistema utilizado en la lista de cumpleaños. Las suscripciones en los clubs suelen efectuarse en un día determinado del año. Este sistema puede ser una desventaja a la hora de atraer nuevos socios pasados varios meses desde la suscripción, ya que los posibles futuros miembros podrían considerar demasiado oneroso tener que pagar la suscripción de un año completo. Sin embargo, que cada socio pague su cuota anual cuando se cumplan los aniversarios de su ingreso tiene la desventaja de que el control manual es complicado. Este tipo de control es sencillo mediante un programa que proporcione mensualmente una lista de suscripciones vencidas que permita conocer las situaciones (¿transitorias?) de impago para efectuar las oportunas reclamaciones. La estructura del fichero sería similar a la de la lista de cumpleaños.

COLECCION DE REVISTAS

Muchas veces, resulta para el coleccionista de revistas un trabajo sumamente laborioso localizar un determinado artículo entre la enorme cantidad de papel sin clasificar. La solución a este problema podría estar en la utilización de SPECFILE. El fichero sería similar al usado en la colección de cassettes. Cada artículo constituiría un registro y no cada revista. Los campos deberían contener la información siguiente: autor, contenido, títulos de la revista, fecha de edición y grado de interés. Es conveniente preparar también un par de campos con observaciones varias sobre cada artículo. El campo contenido deberá codificarse para que sea apto para la clasificación del fichero. Por medio de las rutinas SORT y SEARCH pueden localizarse todos los artículos de un tema determinado, pudiendo escribir después una lista que los reseñe.

FIESTAS

Cuando se dan muchas fiestas, cada vez es más difícil recordar qué invitados estuvieron, qué se sirvió y si la fiesta estuvo bien. Si las fiestas llegasen a ser parte del trabajo de una persona, surgirían problemas que no tienen los anfitriones ocasionales. Los clientes quieren que se recuerden las cosas que les agradan y las que no. Un fichero adecuado ayudará al éxito de las fiestas. Los campos que podrían usarse son: nombre, empresa, fecha, plato principal,

postre. Deberían añadirse varios campos largos donde recoger datos sobre otros invitados y la forma en que se relacionan, aficiones, familia, prejuicios, ideas políticas, alergias alimenticias, etc.

CURRICULUM VITAE

En algún momento de sus vidas la mayoría de la gente tiene que presentar su 'curriculum vitae'. Frecuentemente es difícil recordar ciertos detalles como la fecha en que se aprobó cierto examen, cambios de trabajo, etc. SPECTEXT puede utilizarse para escribir el documento. De este modo, el documento podrá manipularse para insertar nueva información cuando sea necesario. Con SPECMERGE puede enviarse una carta a todas las empresas que se desee. Así se puede dedicar el mínimo tiempo a la escritura del curriculum y quedar más libre para la búsqueda propiamente dicha.

ESTUDIO Y PROCESAMIENTO DE TEXTOS

Cualquier modificación en un trabajo escrito a mano significaría la repetición de todo o parte del mismo. Por el contrario utilizando SPECTEXT, las modificaciones requieren un mínimo esfuerzo y los estudiantes se sienten motivados cuando ven las ventajas que el sistema les ofrece para escribir sus trabajos. Por otra parte la calidad de los trabajos presentados por los universitarios aumenta considerablemente si utilizan un procesado de textos. Los niños deberían comenzar con SPECPRO para pasar después a manejar SPECTEXT cuando hayan adquirido suficiente soltura.

ENTREVISTAS DE TRABAJO

Los anuncios ofreciendo trabajo suelen desembocar en un diluvio de peticiones, con lo que la selección del candidato ideal para el puesto no es tarea fácil. Podríamos ayudarnos en este proceso de un fichero que contuviera como campos las características importantes y que recibiera un registro después de cada entrevista. Finalizadas las entrevistas, se clasificarían las listas en orden creciente de importancia de los campos. Puede suceder que el mejor candidato no sea el primero de la lista, pero sí uno de los primeros. En el caso de que hubiéramos añadido un par de campos largos adicionales para recoger otra información de interés,

posiblemente la selección se haría entre los candidatos que ocupasen los primeros lugares y en base a la información contenida en dichos campos.

TRABAJO DE REPRESENTANTES

El trabajo de un representante podría ayudarse de un fichero similar al que se creó para el primer ejemplo de este capítulo para el trabajo pendiente. Podría usarse SPECMERGE para escribir los informes de los distintos pedidos y el propio fichero serviría como un registro de un viaje de ventas. Estos programas serían de enorme utilidad a aquellos representantes que vivan lejos de las oficinas de la compañía. Porporcionan la mayor parte de las facilidades que normalmente dan las oficinas sin tener que esperar a que se mecanografíen los informes o a recibir el correo. Lo único que se necesita es una impresora de calidad. El último ejemplo de este capítulo tiene que ver con la conexión de una impresora a la computadora que dé al usuario los beneficios de la oficina en casa.

CONEXION DE UNA IMPRESORA

Existen varios interfaces que, puestos en la parte trasera de la computadora, permiten que se conecte una impresora. Es imposible mencionar aquí todos ellos y además sería necesaria una modificación de los programas para trabajar con cada uno. Se ha elegido el interface Centronics de Kempston Micro Electronics para ilustrar la modificación del programa que sería preciso realizar. Hemos seleccionado este interface debido a que con él el comportamiento de una impresora es similar al de la impresora ZX. Pueden usarse los comandos normales LPRINT y LLIST para el manejo de la impresora. La limitación de 32 caracteres por línea se salva fácilmente, pudiendo obtenerse las máximas longitudes de línea posibles.

La posibilidad de elección de impresoras es muy grande. Han tenido gran éxito las impresoras Epson y Microline 80. Silver Reed comercializa un interface que permite manipular con la computadora su máquina de escribir eléctrica. Las máquinas de escribir tienen el inconveniente de su lenta velocidad pero tienen otras ventajas. La primer ventaja es el doble uso que se le puede dar a la máquina como impresora y como máquina de escribir. La segunda ventaja es que la calidad de la impresión que se obtiene con el sistema de tipos es muy superior a la del sistema de matriz

de puntos. La decisión probablemente se tomará habida cuenta del volumen de trabajo que ha de desarrollarse. Los trabajos muy voluminosos requerirán la velocidad de una impresora de matriz de puntos, pero en la mayor parte de las ocaciones la máquina de escribir electrónica será suficiente. Un pequeño factor que podría ser de utilidad para tomar la decisión es que las impresoras tienen sensores que les impiden escribir en los últimos cinco o seis centímetros de una hoja.

LA VERSION DE SPECTEXT PARA IMPRESORA

Sólo se necesitan cambiar 10 líneas del código del procesador de textos para que el interface Kempston pueda manipular una impresora. La mayor parte de los cambios consisten en sustituir el comando PRINT por LPRINT. He aquí los cambios a efectuar.

Línea 20 cambiar PRINT por LPRINT
Línea 21 borrar
Línea 22 borrar
Línea 3020 LET y=x5-1
Línea 3210 cambiar ambos PRINT por LPRINT
Línea 3220 cambiar ambos PRINT por LPRINT
Línea 3230 LPRINT TAB 5; LET y=x5-6: GO TO 3030
Línea 3310 cambiar ambos PRINT por LPRINT
Línea 3400 borrar 'OR p=0'
Línea 3410 cambiar las instrucciones PRINT por LPRINT y TAB por 'TAB x5/2;'
Línea 3420 insertar 'INPUT b$:' antes de 'RETURN'

El último cambio es un poco más extenso que los otros. La línea 6010 debe quedar como sigue:

Línea 6010 LET te=35000: LET li=34000: LET x3=22: LET p=0: IF b$="y" THEN LET p=1: INPUT "HOW MANY LINES/PAGE ";x3: PRINT x3,: INPUT "HOW MANY COLUMNS ";5: PRINT x5' ' "KEY C IF CORRECT": INPUT b$: IF b$ < > "c" THEN CLS: GO TO 60000

El programa funcionará ahora con la impresora ZX o con el interface Kempston y cualquier otra impresora Centronic. El 'INPUT b$' de la línea 3420 es para que se pueda cambiar el pa-

pel cuando la página esté completa. Este cambio podría evitarse utilizando papel continuo. La mayor diferencia que se percibirá con estas modificaciones es que el texto no se imprime en la pantalla. La mayor longitud de la línea haría que la imagen de la pantalla pareciera muy desordenada.

APENDICE

MAPA DE MEMORIA DEL CODIGO MAQUINA

Nombre de la rutina	*Dirección del principio*	*Argumentos de la rutina*	*Dirección de los argumentos*
MOVE	60000 (EA60)	Número de bytes	60001 (EA61)
(abajo)		Destino	60004 (EA64)
		Origen	60009 (EA69)
MOVE	60014 (EA6E)	Número de bytes	60015 (EA6F)
(arriba)		Destino	60018 (EA72)
		Origen	60023 (EA77)
FIND	60028 (EA7C)	Posición en texto	60135 (EAE7)
		Posición en cadena	60137 (EAE9)
		Fin del texto	60139 (EAEB)
		Fin de la cadena	60141 (EAED)
		Bandera	60143 (EAEF)
		Contador secundario	60145 (EAF1)
FILL	60151 (EAF7)	Primer byte	60152 (EAF8)
		Ultimo byte	60155 (EAFB)
		Código de carácter	60158 (EAFE)
SORT	60185 (EB19)	Siguiente registro	60173 (EB0D)
		Número de campo	60182 (EB16)
		Bandera	60183 (EB17)
DUMP	60500 (EC54)	Número de campos	60495 (EC4F)
		Siguiente carácter	60496 (EC50)
		Dirección puntero	60498 (EC52)
START	60570 (EC9A)	Fin del fichero	60567 (EC97)
		Posición del texto	60135 (EAE7)
		Fin del registro	60565 (EC95)
		Bandera	60569 (EC99)

Las rutinas del código máquina utilizan otras posiciones de memoria entre los bloques para el almacenamiento temporal de la información. Este bloque de código ocupa 645 bytes a partir de la posición 60000 y no puede reubicarse en otro lugar. Se ha dejado espacio por encima del código para que sean posibles posteriores adiciones y también para uso de la rutina SORT que puede necesitar mucho espacio para el almacenamiento temporal de registros.

MAPA DE MEMORIA DE LOS PROGRAMAS SPECTEXT Y SPECFILE

Hasta la posición 33999	Area del programa en BASIC
Desde 34000 a 34997 (84D0) a (88B5)	Buffer de línea, área para el almacenamiento temporal del texto durante la edición, muy usada por SPECFILE para contener la información de la estructura del fichero
Desde 34998 a 59999 (88B6) a (EA5F)	Buffer del texto, zona principal de almacemiento de ambos programas, limitada para impedir que se 'machaque' el código máquina
Desde 60000 a 60645 (EA60) a (ECE5)	Espacio para nuevo código máquina
Desde 60646 al final (ECE6) a (FFFF)	Este espacio se utiliza también por la rutina SORT para almacenar temporalmente registros, pudiendo llegar a ocupar hasta 600 bytes desde la posición 62000

Programas de la vida real

El Spectrum de 48K puede utilizarse como un potente procesador de textos en los negocios, la educación y en los usos domésticos. Puede manipular la impresora ZX, pero también impresoras de matriz de puntos y de margarita por medio de un adaptador (interface). Este libro contiene técnicas de programación, así como algunos paquetes de programas comerciales.

El procesador de textos Spectrum

El listado de los programas se complementa con una descripción detallada de las técnicas de diseño de procesadores de textos y sistemas de ficheros. La mayor parte de los programas está en BASIC. Cuando era necesario se han usado conjuntamente el lenguaje máquina y el BASIC, utilizando una aproximación innovadora al lenguaje máquina que consiste en comprobar y corregir el programa mediante unos equivalentes BASIC de las instrucciones en ensamblador antes de proceder a su ensamblado final.

PUBLICACIONES SOBRE EL ZX SPECTRUM (TS 2068)

ISBN: 968-451-731-9

